Guida per l’Utente del C64

TRADOTTA DA MAC (MARCO CATENI) PER EDICOLAC64

Benvenuto sul Progetto 64!

Lo scopo del Progetto 64 è di preservare documenti in formato testo elettronico del Commodore 64 che, altrimenti, potrebbero cessare di esistere a causa del rapido progresso della tecnologia per computer che ha fatto diminuire l'interesse verso i computer ad 8 bit.

Per preservare i contenuti di documernti originali sono stati fatti enormi sforzi, ma certe parti come i diagrammi, i listati di programma e gli indici possono essere/sono stati modificati o sacrificati a causa delle limitazioni degli ormai obsoleti file di testo. I diagrammi possono esser stati eliminati dove non era più possibile mettere i codici ASCII. I listati di programma possono aver perso codici di visualizzazione dove non è stata possibile alcuna sostituzione. Nelle tabelle dei contenuti e negli indici possono esser stati cambiati i riferimenti al numero di pagina con riferimenti al numero di paragrafo. Si prega di accettare le nostre scuse per queste limitazioni, modifiche ed eventuali omissioni.

I nomi dei documenti sono limitati alla convenzione dei file DOS di 8+3 caratteri. I primi caratteri del nome del file è una abbreviazione del nome originale del documento. Segue il numero della versione del testo elettronico. Dopo questo può esserci una lettera per indicare il particolare sorgente del documento. Infine, al documento viene data l’estensione .TXT.

Testo elettronico del Progetto 64 del “Manuale per l'Utente del Microcomputer Commodore 64 (seconda edizione)~

convertito in testo elettronico da Frank Jeno Kontros jeno@kontr.uzhgorod.ua .

C64UG210.TXT, Giugno 1997, testo elettronico #251#

tradotto in lingua italiana da Marco Cateni voltiars@gmail.com

GuidaUtenteC64.doc, Gennaio 2011, documento ipertestuale

Manuale per l'Utente del MicroComputer Commodore 64

c o m m o d o r e
C O M P U T E R

GUIDA ALL'UTENTE PER IL COMMODORE 64

Pubblicato da Commodore Business Machines, (UK) Ltd.

INDICE

INTRODUZIONE

Il tuo nuovo COMMODORE 64 è il miglior computer domestico disponibile al momento. Puoi utilizzare il tuo COMMODORE 64 per ogni cosa: dalle applicazioni commerciali ai fogli di calcolo per la gestione interna, ai giochi eccitanti. Il 64 ti offre molta memoria (64K), molti colori (16 colori diversi), molti suoni (musica ed effetti sonori) e molto divertimento ed usi pratici. Puoi utilizzare software preconfezionato oppure puoi scrivere i tuoi programmi personali in un BASIC facile da imparare.

La guida all'utente, facile da leggere, contiene tutte le informazioni necessarie per configurare a modo la tua apparecchiatura, per capire come rendere operativo il tuo nuovo COMMODORE 64 ed imparare a creare i tuoi propri semplici programmi in BASIC.
La guida all'utente è intesa per introdurti al computer, ma è al di là dello scopo di questo manuale dirti tutto ciò che devi conoscere sui computer o sul BASIC. Questa guida si rivolge a te come principiante dato che gli argomenti qui presentati si trovano su una varietà di pubblicazioni che li spiegano con maggiori particolari.
Per coloro che non vogliono imparare la programmazione, non hanno bisogno di cercare per tutto il libro come utilizzare i programmi ed i giochi preconfezionati per il Commodore, od altro software preconfezionato da terzi. Tutte le informazioni che devi sapere le abbiamo messe proprio davanti ai Capitoli 1 e 2.

Molte funzioni eccitanti sono in tua attesa dentro il tuo COMMODORE 64. Il tuo nuovo computer ti dà la miglior grafica dell'industria per microcomputer, che noi chiamiamo FIGURE GRAFICHE. Le Figure Grafiche ti permettono:

- Disegnare le tue immagini personali in diversi colori, proprio come quelli che vedi sui giochi a schermo di tipo arcade.
- Animare fino ad 8 diversi livelli d'immagine alla volta.
- Muovere le tue creazioni per tutto lo schermo.
- Raddoppiare la loro dimensione.
- Passare le immagini davanti o dietro a ogni altra.
- Usare la rilevazione automatica di collisione che fa sapere al computer di fare quello che vuoi quando le figure si urtano fra di loro.

Queste funzioni ti permettono di disegnare i tuoi propri giochi. Il COMMODORE 64 ha al suo interno anche effetti musicali e sonori che competono con molti dei ben noti sintetizzatori musicale. Questa caratteristica del tuo computer ti dà:

- 3 voci indipendenti, ognuna con un campo di 9 ottave tipo pianoforte.
- 4 forme d'onda diverse (dente di sega, triangolare, impulso variabile e rumore).

- Un generatore d'inviluppo programmabile ADSR (attacco, decadimento, sostentamento e rilascio).
- Un filtro programmabile per passa-alto, passa-basso e passa-banda che puoi utilizzare per ciascuna voce.
- Controlli variabili di risonanza e volume.

Se vuoi riprodurre la tua musica in modo professionale, il COMMODORE 64 ti permette di collegare l'uscita audio con quasi qualsiasi sistema di amplificazione in alta-qualità.

Via via che il tuo calcolatore avrà bisogno di crescere, il tuo sistema può farlo. Puoi espanderlo collegando il tuo COMMODORE 64 ad altre apparecchiature, note come unità periferiche. Questi accessori comprendono elementi come questi:

- Il registratore DATASSETTE*, per i nastri.
- Il gestore disco VIC 1541 (fino a cinque contemporaneamente).
- Le stampanti a matrice di punti del COMMODORE, per pure e semplici copie dei tuoi programmi, lettere, ecc.
- La cartuccia MODEM, per accedere attraverso il tuo telefono alle massicce basi di dati di computer più grandi, come pure i servizi di centinaia di specialisti ed un varietà di reti informatiche.
- Il monitor a colori Commodore 1701.

Se possiedi già un gestore disco VIC 1540, il tuo commerciante può aggiornarlo per il COMMODORE 64.
La Commodore desidera che il tuo nuovo Commodore 64 ti faccia veramente divertire. E per divertirsi, tieni a mente che per imparare la programmazione ci vuole tempo per cui, mentre leggi la GUIDA ALL'UTENTE, cerca di avere pazienza. Però, prima di iniziare, per favore perdi qualche minuto per compilare ed impostare la scheda di proprietà/registrazione che accompagna il tuo computer. Questa ti assicurerà che il tuo COMMODORE 64 sia registrato a modo dalla sede centrale della Commodore e possa ricevere le informazioni più aggiornate riguardanti le prossime migliorìe per la tua macchina.

NOTA: Mentre viene presentato questo manuale, sono in sviluppo molti programmi. Per favore tieni d'occhio il tuo commerciante locale Commodore, i Periodici per Utenti Commodore ed i Club che ti terranno aggiornato sulla quantità di programmi applicativi da scrivere per il COMMODORE 64, a livello mondiale.

* DATASSETTE è un marchio commerciale registrato della Commodore Business Machines, Inc.

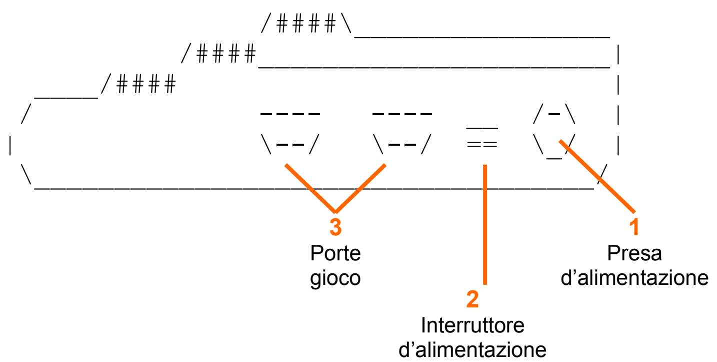
1. CONFIGURAZIONE

1.1. DISIMBALLAGGIO E COLLEGAMENTO DEL 64

Le seguenti istruzioni passo-passo ti mostrano come collegare il 64 al tuo televisore, al sistema audio od al monitor ed assicurarti che tutto funzioni correttamente.
Prima di collegare qualcosa al computer, controlla il contenuto della confezione del 64. Oltre a questo manuale dovresti trovare i seguenti oggetti:

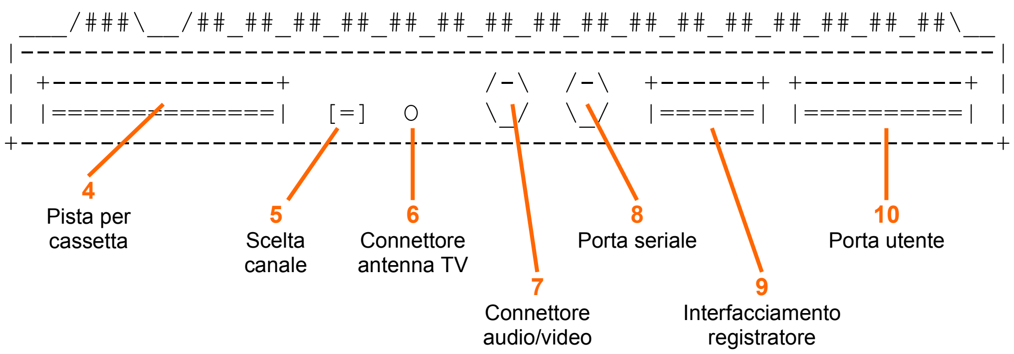
1. Il Commodore 64
2. L'alimentatore (una scatola nera con una spina per AC ed il cavo d'alimentazione)
3. Il cavo per il video

Se manca un qualsiasi elemento, ricontrolla immediatamente con il tuo fornitore per una eventuale sostituzione.
Prima di tutto dai un'occhiata alla disposizione dei vari collegamenti sul computer e cosa fanno.



COLLEGAMENTI DEL PANNELLO LATERALE

1. PRESA D'ALIMENTAZIONE. L'estremità libera del cavo d'alimentazione viene collegata qui per fornire l'alimentazione al 64.
2. INTERRUTTORE D'ALIMENTAZIONE. Attiva l'alimentazione al 64.
3. PORTE GIOCO. Ogni connettore del gioco può accettare un joystick o paddle di controllo del gioco, mentre la penna ottica può essere collegata solo nella porta di gioco più vicina alla parte anteriore del tuo computer.



COLLEGAMENTI POSTERIORI

4. CARTUCCIA. La fessura rettangolare a sinistra accetta programmi o cartucce di gioco.
5. SELETTORE DI CANALE. Usa questo interruttore per sceglire il canale TV sul quale verranno visualizzate le immagini del computer.
6. CONNETTORE ANTENNA TV. Questo connettore fornisce sia le immagini che i suoni alla tua televisione.

7. USCITA AUDIO & VIDEO. Questo connettore fornisce un audio diretto, che può essere collegato a unsistema audio di alta qualità, ed un segnale video composito che può essere portato ad un televisione o monitor, come il monitor a colori Commodoro 1701.
8. PORTA SERIALE. Puoi collegare una stampante COMMODORE. Attraverso questo connettore puoi collegare direttamente al Commodore 64 anche un singolo gestore-disco VIC 1541.
9. INTERFACCIA REGISTRATORE. Si può collegare al computer un registratore DATASSETTE in modo da poter salvare le informazioni su nastro per riutilizzarle in seguito.
10. PORTA UTENTE. Alla porta utente si possono collegare varie cartucce d'interfacciamento, come il MODEM o la cartuccia di comunicazione RS-232.

1.2. INSTALLAZIONE

COLLEGAMENTI AL TUO TV

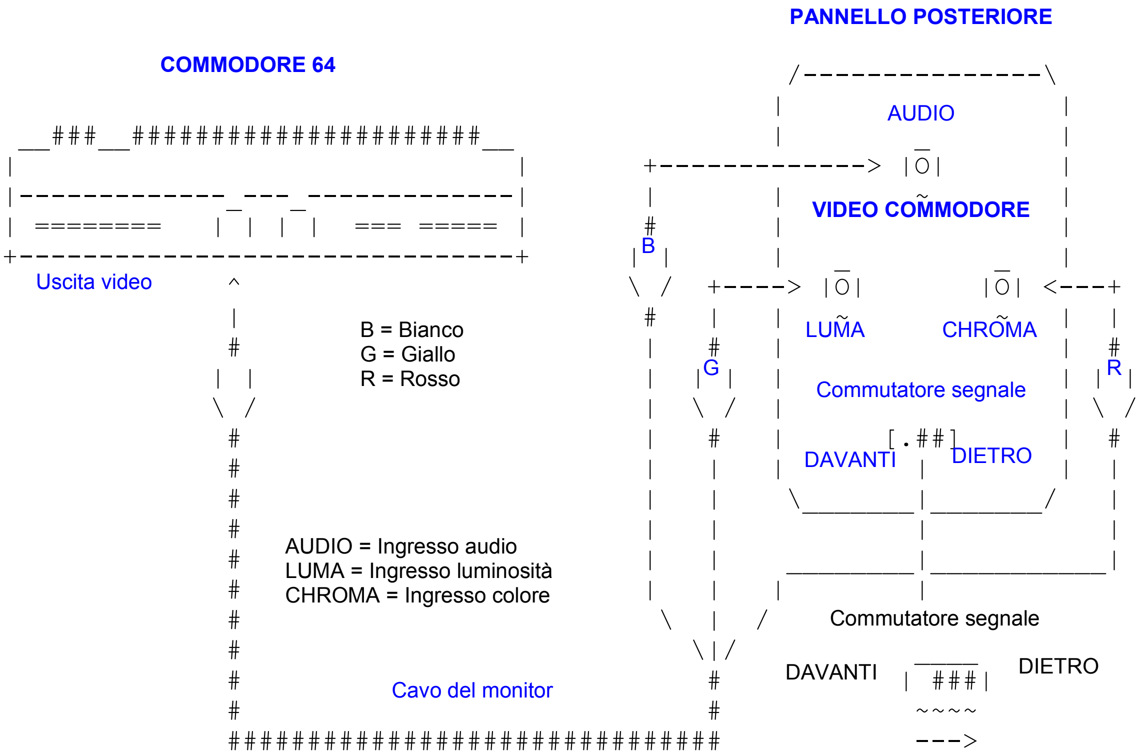
Collega il computer alla tua TV come mostrato qui sotto.

```
                                           +--+-----------------+--+
                                           |  | /-------------\#|  |
               /---------------------------|  | |             |#|  |
               |                           |  | |             |#|  |
               |                           |  | |             | |  |
               |                           |  | |             | |  |
               |                           |  | \-------------/ |  |
               |                           +==+=================+==+
               |
 /-------------------------\
|                           |=---\
| ####################  ## |     |
| ####################  ## |     |
|  ##################   ## |     |
|    ##############     ## |     |
 \_________________________/     |
                                 |
                                 |
                                 |
                               +----+
                               |    |  Alimentatore
                               |    |
                               +----+
                                 |
                                 |
```

1) Collega un'estremità del cavo TV alla presa del segnale TV di tipo phono sul retro del 64. Basta spingerla. Si può utilizzare una estremità qualsiasi del cavo.
2) Collega l'altra estremità del cavo alla presa per l'antenna. Basta spingerla.
3) Collega il cavo dell'alimentatore nella presa di corrente sul lato del Commodoro 64. Basta spingerla. La presa è "munita di chiave" per permettere l'inserimento solo in un modo, per cui non puoi collegare il cavo dell'alimentatore nel modo sbagliato. L'alimentatore converte la corrente di casa nella forma utilizzata dal computer.

Adesso il 64 è collegato correttamente. Utilizzando la tua TV non sono necessari ulteriori collegamenti.

COLLEGAMENTI AL MONITOR 1701



1.3. COLLEGAMENTI FACOLTATIVI

Poiché il 64 fornisce un canale audio ad alta fedeltà, potresti volerlo eseguire attraverso un amplificatore di qualità per rendere il suono al meglio possibile. Oltre ciò, il 64 fornisce anche un segnale standard, video composito, che può essere passato ad un monitor TV.
Queste possibilità si trovano sulla presa d'uscita audio/video sul pannello posteriore del 64. Il modo più facile per avere accesso a questi segnali è di utilizzare un cavo audio standard con connettore DIN a 5 pin (non fornito). Questo cavo si collega direttamente al connettore audio/video del computer. Due dei quattro pin, all'estremità opposta del cavo, portano i segnali audio e video. Puoi costruire da te il cavo usando lo schema dei pin mostrato nell'Appendice I della guida.
Di solito il filo NERO del cavo DIN fornisce il segnale AUDIO. Questa spina può essere collegata all'entrata AUSILIARE di un amplificatore o al connettore AUDIO IN di un monitor od altro sistema video, come un registratore di cassette video (VCR).
Il filo BIANCO o ROSSO di solito fornisce il segnale VIDEO diretto. Questa spina viene collegata al connettore VIDEO IN del monitor o al settore d'ingresso video di qualche altro sistema video, come un VCR.
In base al costruttore del tuo cavo DIN, il codice dei colori dei piedini può essere differente. Se nonostante i collegamenti suggeriti non ricevi un segnale audio o video, per la piedinatura serviti dello schema dei pin mostrato nell'Appendice I.

```
                __###__################################__
               |              USCITA AUDIO/VIDEO           |
               |                                           |
               |-----------------_---_--------------------|
               | =========   = o |‾| |‾|   =====  ========= |
               +-------------------------------------------+

                                   ^
                                   |

                                  /-\
                                  | |
                                  | |
                                   #
                                   |
                                   |
                    ALL'INGRESSO   |
                         AUX       |
                          O        |
                        RADIO      |      ALL'INGRESSO VIDEO
           _____________          / \
          /             \
+----++-------------++-----+  <-#-/   \-#->  +-------------+----+
|    || ###  ====== ||     |                 | /---------\ | O  |
|    || ===   OoooO ||     |                 ||           || =  |
|    ||-------------||     |                 ||           || o  |
|    ||  |   |   |  ||     |                 | \_________/ |    |
|    ||  |   |   |  ||     |                 +-------------+----+
+----++=============++-----+                        MONITOR TV
        SISTEMA Hi-Fi
```

Se hai acquistato qualche apparecchiatura periferica, come un gestore disco VIC 1541, una stampante MPS 801, 802 o 803, un plotter 1520 o un monitor 1701, potresti collegarli tutti insieme. Ogni apparecchiatura comprende un manuale dell'utente che mostra il giusto procedimento per collegarsi al computer: consulta questi manuali.

Un sistema completo potrebbe essere simile a questo.

[Immagine omessa]

1.4. OPERAZIONE

USO DEL 64

Attiva il computer premendo l'interruttore sul pannello di destra guardando il computer di fronte. Dopo qualche attimo sullo schermo della TV verrà visualizzato quanto segue:

```
**** COMMODORE 64 BASIC V2 ****
  64K RAM SYSTEM  38911 BASIC BYTES FREE

  READY.
_
```

IL CURSORE INDICA CHE IL COMMODORE 64 STA ASPETTANDO UNA TUA IMMISSIONE

Se la tua TV ha una manopola per la sintonizzazione fine, regola la TV fino ad ottenere un'immagine chiara.
Per una visualizzazione migliore, sul televisore potresti regolare anche i controlli per il colore e per la tinta. Per avere il tutto configurato a modo, puoi usare la procedura per la regolazione del colore descritta più avanti. Quando ottieni la prima immagine, la schermata dovrà apparire per lo più blu scuro con un bordo e le lettere blu chiare.

Se non ottiene questi risultati, ricontrolla i cavi ed i collegamenti. Il diagramma proposto sopra ti aiuterà ad isolare ogni problema.

1.5. TABELLA PER LA LOCALIZZAZIONE DEI PROBLEMI

Sintomo	Causa	Rimedio
La luce dell'indicatore non è accesa	Il computer non è acceso	Assicurarsi che l'interruttore di alimentazione sia in posizione "On"
	Il cavo di alimentazione non è collegato	Controllare che il cavo non sia scollegato dalla presa di alimentazione
	L'alimentatore non è collegato	Controllare che il cavo dell'alimentatore sia collegato al computer
	Fusibile bruciato nel computer	Far sostituire il fusibile da un fornitore autorizzato
Manca l'immagine	TV su canale errato	Controllare altri canali (3 o 4)
	Collegamento errato	Il computer non è collegato all'antenna VHF
	Cavo video non collegato	Controllare il collegamento del cavo all'uscita TV
	Computer impostato su un canale errato	Impostare il computer sullo stesso canale TV (3 o 4)
Immagini casuali sul TV con una cartuccia inserita	Cartuccia non inserita a modo	Reinserire la cartuccia dopo aver spento il computer
Immagini senza colori	Scarsa sintonizzazione sul TV	Risintonizzare il TV
Immagini a colori sbiaditi	Regolazione errata sul TV	Regolare i controlli per il colore/brillantezza sul TV
Suoni con eccessivo rumore di fondo	Volume TV troppo alto	Regolare il volume del TV
Immagini a posto, ma non i suoni	Volume TV troppo basso	Regolare il volume del TV
	Collegamento errato sull'uscita Aux	Collegare il jack audio all'ingresso Aux di un amplificatore e selezionare l'ingresso Aux

INFORMAZIONE: Il 64 è stato progettato affinchè lo possano utilizzare tutti. Tuttavia, noi della Commodore, riconosciamo che gli utenti di computer, di quando in quando, possano imbattersi in qualche difficoltà. Per aiutarti a risponde alle tue domande ed a darti alcune idee sulla programmazione, la Commodore ha creato diverse pubblicazioni per aiutarti.
Potresti anche ritenere essere una buona idea collegarsi ad un club di Utenti Commodore per incontrare qualche altro proprietario del 64 che possa aiutarti a fare esperienza.

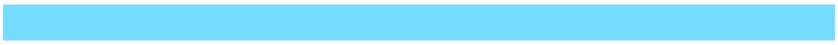
CURSORE

Il blocchetto lampeggiante sotto READY si chiama cursore. È un indicatore che mostra dove e cosa stai digitando sulla tastiera e che cosa verrà visualizzata sullo schermo. Via via che digiti, il cursore si sposterà in avanti di un posto in modo che la posizione d'origine del cursore venga sostituita dal carattere che hai digitato. Prova a digitare qualcosa sulla tastiera e vedrai il cursore spostarsi mentre i caratteri digitati verranno visualizzati sullo schermo.

1.6. REGOLAZIONE DEL COLORE

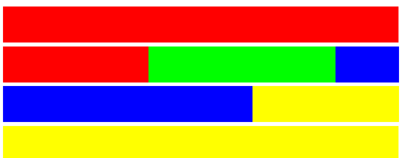
Per ottenere un modello di colori sul monitor, in modo da poter facilmente regolare la serie, c'è un modo semplice. Anche se ancora non hai familiarità con le operazioni del computer, basta seguire quanto descritto avanti per vedere la facilità di utilizzo del tuo computer.
Per prima cosa guarda sul lato sinistro della tastiera ed individua il tasto contrassegnato <CTRL>. Questo sta per ConTRollo e viene utilizzato, assieme ad altri tasti, per informare il computer di svolgere un compito speciale. Per utilizzare una funzione di controllo, tieni premuto il tasto <CTRL> mentre premi un altro tasto.
Prova questo: tieni premuto il tasto <CTRL> mentre premi anche il tasto <9>. Quindi rilascia entrambi i tasti. A prima vista sembra che non sia successo niente di evidente, ma se adesso premi un tasto qualsiasi, lo schermo mostrerà il carattere in modo inverso invece che in modo normale -- come il messaggio d'apertura o qualsiasi altra cosa digitata in precedenza.
Tieni premuta la <BARRA DELLO SPAZIO>. Cosa succede? Se hai eseguito correttamente la procedura sopra descritta, dovresti vedere una barra blu chiara che si sposta attraverso la schermata per poi portarsi in basso, sulla riga successiva, per tutto il tempo che tieni premuta la <BARRA DELLO SPAZIO>.

READY.



Adesso tieni premuto <CTRL> mentre premi uno qualsiasi degli altri tasti numerici. Ognuno di essi riporta un colore contrassegnato sulla parte anteriore. Qualsiasi cosa visualizzata da questo momento sarà in quel colore. Per esempio, tieni premuto <CTRL> ed il tasto <8> e rilasciali entrambi. Adesso tieni giù la <BARRA DELLO SPAZIO>. Osserva la visualizzazione. Adesso la barra è gialla! In maniera analoga puoi cambiare la barra con uno qualsiasi dei colori indicati sui tasti numerici tenendo premuto <CTRL> insieme al tasto appropriato.
Modifica la barra con altri colori e poi regola i controlli per il colore e la tinta sul tuo monitor in modo che la visualizzazione sia uguale ai colori che hai scelto.
La visualizzazione dovrà apparire come qualcosa di simile:

READY.



Barra rossa <3>
Barre rossa <3>, verde <6> e blu<7>
Barre blu <7> e gialla<8>
Barra gialla <8>

A questo punto ogni cosa è regolata bene e lavora correttamente. I capitoli successivi ti introdurranno al linguaggio BASIC. Comunque, puoi iniziare subito ad utilizzare alcune delle molte applicazioni e dei giochi prescritti disponibili senza per questo dover conoscere la programmazione del computer. Ciascuno di questi pacchetti contiene le informazioni dettagliate su come utilizzare il programma. È comunque consigliabile che tu esamini i primi pochi capitoli di questo manuale per avere più familiarità con le operazione del tuo nuovo sistema.

1.7. ESPANSIONE DEL TUO SISTEMA CON UNITÀ PERIFERICHE FACOLTATIVE

La Commodore offre una vasto gamma di dispositivi periferici che espandono le possibilità del tuo computer. Queste unità periferiche comprendono:

- ✓ dispositivi di memorizzazione
- ✓ stampanti e plotter
- ✓ monitor
- ✓ modem per telecomunicazioni
- ✓ aggiunte per il di gioco
- ✓ moduli grafici e vocali
- ✓ controllori di scrivania

DISPOSITIVI DI MEMORIZZAZIONE

Gestori-disco

I gestori-disco della Commodore ti permettono di memorizzare ampie quantità di informazioni su dischetti da 5-1/4". Tali dischetti offrono una veloce memorizzazione e ricerca, ed automaticamente tengono traccia di tutti i tuoi file in una cartella, o indice, che puoi visualizzare sul tuo schermo o stampare su una stampante.

Oltre ciò, puoi aggiungere altri gestori disco mettendoli a catena aul tuo computer. Mettere a catena significa collegare un gestore-disco al computer, e poi collegarne altri in sequenza.

Acquistando la Scheda d'Espansione per l'Interfaccia IEEE del Commodore 64, puoi anche collegare qualsiasi gestore-disco IEEE come il CBM 8050 della Commodore oppure o il 4040 Dual Floppy Disk Drive.

Il capitolo 2 comprende informazioni dettagliate sull'uso dei gestori-disco.

DISPOSITIVI DI STAMPA E DI TRACCIAMENTO

Stampanti

Sul 64 puoi collegare le stampanti della Commodore. Queste stampanti a matrice di punti sono abbastanza economiche. Acquistando la Scheda d'Espansione per l'Interfaccia IEEE del Commodore 64, puoi anche collegare qualsiasi stampante IEEE come la stampante a qualità di lettere Commodore 6400 o la stampante ad alta velocità 8023 a matrice di punti.

Stampante/Tracciatore grafico

La stampante/plotter 1520 della Commodore stampa e traccia grafiche in quattro colori (nero, blu, rosso e verde). Con la 1520 puoi disegnare diagrammi a barre, a torta ed una varietà di grafiche complesse.

IL MONITOR 1701/1702

Il monito a colori a 14" della Commodore offre un'immagine superiore a colori, in alta risoluzione, che valorizza la tua esperienza sul computer. Questo monitor può essere collegato al 64 con un cavo DIN ad 8 pin. La Guida all'Utente del Monito A colori 1701/1702, che accompagna il monitor, spiega chiaramente tutti i collegamenti. Per lo schema dei piedini del connettore ad 8 pin puoi anche consultare l'Appendice I.

AGGIUNTE PER IL GIOCO ED ALTRI IMPIEGHI

La Commodore offre joystick e paddle che migliorano la giocabilità sul tuo computer. Queste aggiunte hanno anche altre applicazioni. Una di queste possibilità è la penna ottica della Commodore che, con il software adatto, permette la comunicazione tra computer e schermo.

AIUTI PER LA GRAFICA DELLA COMMODORE

La Commodore fornisce una varietà di aiuti per la programmazione della grafica: il SIMON'S BASIC che aggiunge 114 nuovi e potenti comandi al BASIC incluso l'aiuto alla programmazione e comandi per la grafica; ed il LOGO, un linguaggio di programmazione facile da imparare con grafica TURTLE.

AGGIUNTE PER LA MUSICA

Molto presto la Commodore offrirà una Tastiera Musicale ed una aggiunta 3-pad per le percussioni chiamata Digi-drum. Tutti e due i prodotti comprenderanno particolari pacchetti di software. Queste aggiunte incrementeranno le capacità di creare musica del computer 64.

COLLEGAMENTO VERSO UNO STEREO

Le possibilità di creare audio e musica del COMMODORE 64 possono essere valorizzate collegando il computer a un amplificatore di alta qualità ed altoparlanti stereofonici. Il cavo DIN ad 8 pin, che abbiamo visto nel paragrafo che riguardava il Monitor a Colori 1701/1702, può essere utilizzato anche per collegare il tuo computer ad un amplificatore.

PROGETTARE UN SISTEMA COMPUTER PER LE PROPRIE NECESSITÀ

La Commodore offre una varietà di unità periferiche che ti permettono di creare il sistema che più ti aggrada. Noi offriamo diversi tipi di dispositivi per la memorizzazione, la stampa e la telecomunicazione; puoi scegliere quello che per te va meglio. Per maggiori informazioni sulle unità periferiche del Commodore, leggi la Guida alle Unità Periferiche del Commodore ed i periodici del Commodore (discussi nell'Appendice H) e consulta il tuo fornitore di Commodore.

2. PER INIZIARE

2.1. COMUNICAZIONE CON IL TUO 64: LA TASTIERA

La tastiera del computer ti permette di comunicare con il tuo 64. Usa i tasti per dire al computer quello che vuoi fargli fare e per rispondere alle domande che il computer visualizza sullo schermo. La tastiera è simile ad una normalissima macchina da scrivere, ma il computer ha anche dei tasti speciali che permettono al 64 di fare qualcosa in più di una macchina da scrivere. Quando leggi le prossime poche pagine, dai un'occhiata a questi tasti speciali.

< RETURN> — Il tasto <RETURN> (o <INVIO>) dice al computer di esaminare quello che hai digitato e di mettere in memoria quelle informazioni.
Il tasto <RETURN> fa anche in modo che il cursore si porti sulla riga successiva.

NOTA: La memoria comprende tutte le informazioni in possesso del computer senza doverti richiedere dove guardare.

< SHIFT> — Il tasto <SHIFT> lavora come il tasto per le maiuscole su una vera e propria macchina da scrivere: in pratica ti permette di visualizzare lettere maiuscole o i caratteri in alto su quei tasti che riportano un carattere doppio.

```
  ____       ____       ____
 / !  \     / "  \     / #  \
|| 1  ||   || 2  ||   || 3  ||
/\____/\   /\____/\   /\____/\
|______|   |______|   |______|
```

Quando utilizzi la grafica sulla facciata dei tasti, il tasto <SHIFT> visualizza il carattere grafico che si trova sul lato DESTRO del tasto.

```
  ___      ___      ___
 /   \    /   \    /   \
|| Q ||  || W ||  || E ||
/\___/\  /\___/\  /\___/\
|_|_-_|  |_-_|_|  |_|_-_|
```

Quando utilizzi i quattro tasti speciali di funzione, sul lato destro della tastiera, il tasto <SHIFT> ti dà le funzioni sulla FACCIATA del tasto (F2, F4, F6 ed F8).

TASTI CHE TI PERMETTONO DI FARE MODIFICHE

<CRSR>

Il cursore è il piccolo rettangolo colorato che indica la tua posizione sullo schermo. Ci sono due tasti CuRSoRe:

<CRSR-UP/DOWN> sposta il cursore su e giù
<CRSR-LEFT/RIGHT> sposta il cursore a sinistra ed a destra

Per spostare il cursore in alto devi usare il tasto <SHIFT> insieme al tasto <CRSR-UP/DOWN>, o insieme al tasto <CRSR-LEFT/RIGHT> per spostare il cursore a sinistra.
Per far muovere il cursore più di uno spazio, non c'è bisogno di premere tutte le volte il tasto <CRSR>,.ma basterà tenerlo giù fino a far arrivare il cursore dove vuoi.

< INTS/DEL>

DEL significa DELete (Cancellazione). Quando premi il tasto <DEL>, il cursore si sposta indietro di uno spazio e cancella il carattere che vi si trova.

PRINT "ERROR"#_
PRINT "ERROR"#

Quando vuoi cancellare nel centro di una riga, porta il cursore a sinistra del carattere che vuoi Cancellare.

FIX IT AGAINS, SAM
FIX IT AGAINS_ SAM

Quindi premi il tasto <DEL>, i caratteri a destra si porteranno automaticamente a sinistra chiudendo lo spazio lasciato dal carattere cancellato.

FIX IT AGAIN, SAM

INST significa INSerimenTo. Quando vuoi inserire caratteri su una riga, devi utilizzare il tasto <SHIFT> insieme al tasto <INST/ DEL>.
Se hai dimenticato alcuni caratteri su una riga, usa i tasti <CRSR> per portare il cursore sull'errore.

WHILE U WERE OUT
WHILE _ WERE OUT

In seguito, mentre tieni premuto il tasto <SHIFT>, premi il tasto <INST/ DEL> tante volte quanti sono i caratteri da aggiungere.
<INST> non sposta il cursore, ma aggiunge spazio tra il cursore ed il carattere alla sua destra.

WHILE _ U WERE OUT
WHILE YOU WERE OUT

Usa i tasti <DEL> e <INST> insieme per sistemare i caratteri sbagliati.

WE'RE NUMBER TWO!
WE'RE NUMBER !
WE'RE NUMBER _ !
WE'RE NUMBER ONE!

<CLR/HOME>

HOME riporta il cursore sull'angolo superiore sinistro della schermata. Questa posizione è la "CASA" del cursore (cioè la posizione di partenza).

CLR sta per CLeaR (Pulitura). Quando usi il tasto <SHIFT> insieme al tasto <CLR/HOME>, lo schermo viene pulito (vuotato) ed il cursore ritorna alla posizione di casa.

< RESTORE>

Il tasto RESTORE riporta il computer alla sua condizione normale ripristinando le condizioni preimpostate (per es., il colore della schermata di default è blu, i microcircuiti di I/O per default sono OFF, ecc.). RESTORE agisce su cose come liberare lo schermo, riportandolo al colore d'origine, e disattiva i microcircuiti per la creazione di immagini/suoni.

NOTA: Affinché <RESTORE> funzioni, insieme al tasto <RESTORE> devi tenere premuto il tasto <STOP>.

Per esempio, supponiamo che tu abbia appena eseguito un programma di musica che ti ha anche cambiato la schermata in rosso e giallo mentre LISTava il programma. Al momento che premi <STOP> e <RESTORE>, alla fine del programma, cesserà l'ultima nota dal programma mentre lo schermo ritornerà blu con l'unica cosa visualizzata il prompt READY.

TASTI FUNZIONE

I tasti sul lato destro della tastiera, da F1 a F8, sono tasti funzione che puoi programmare per far loro eseguire una varietà di compiti. La spiegazione dell'istruzione GET, nel Capitolo 5, ti spiega il modo di programmare in BASIC questi tasti.

<CTRL>

I tasto ConTRoLlo ti permette di impostare colori ed eseguire altri compiti particolari chiamati funzioni di controllo.
Per impostare i colori tieni premuto il tasto <CTRL> mentre premi il tasto con il colore che vuoi. Con il tasto <C=> puoi ottenere più di otto colori.
Il capitolo 6 tratta maggiormente sui colori.
Per avere una funzione di controllo, tieni premuto il tasto <CTRL> mentre premi un altro tasto. Le funzioni di controllo vengono normalmente utilizzato in software preconfezionato come un sistema per l'elaborazione di testi.

<RUN/STOP>

Premendo il tasto <STOP> puoi fermare un programma BASIC mentre è in esecuzione. Puoi utilizzare il tasto <STOP> anche per fermare una stampa mentre sta ancora stampando.

<RUN> ti permette di caricare automaticamente un programma da cassetta. Quando vuoi utilizzare il tasto <RUN>, devi farlo con il tasto <SHIFT>.

<C=>

TASTO COMMODORE Il tasto Commodore <C=> può fare due cose:

1. <C=> ti permette di scambiare tra la modalità per la visualizzazione maiuscola e minuscola (le lettere ed i caratteri sulla parte superiore dei tasti) e la modalità per la visualizzazione maiuscola/grafica (le lettere maiuscole e la grafica sulla facciata dei tasti).

Per scambiare le modalità, premi contemporaneamente i tasti <C=> e <SHIFT>.
Quando accendi la prima volta il tuo 64, esso si troverà in modalità maiuscola/grafica, il che significa che tutto ciò che immetti sarà a lettere maiuscole. Quando ti trovi in questa modalità, puoi anche stampare tutti i grafici sulla facciata dei tasti.

- Per stampare i grafici sul lato destro di un tasto, tieni premuto il tasto <SHIFT> contemporaneamente al tasto con la grafica che vuoi stampare. Quando ti trovi in modalità maiuscola/grafica, puoi stampare solo le grafiche sul lato destro.
- Per stampare i grafici sul lato sinistro di un tasto, tiene premuto il tasto <C=> insieme al tasto grafico. In entrambe le modalità puoi stampare la grafica sul lato sinistro.

2. Il tasto <C=> ti permette anche di accedere alla seconda serie degli otto colori alternativi non mostrati sul tasti per il colore. Per ottenere quest'altri colori, tiene premuto il tasto <C=> insieme al numero del colore che vuoi.

<C=> 1 ARANCIO	<C=> 5 GRIGIO 2
<C=> 2 MARRONE	<C=> 6 VERDE CHIARO
<C=> 3 ROSSO CHIARO	<C=> 7 BLU CHIARO
<C=> 4 GRIGIO 1	<C=> 8 GRIGIO 3

2.2. CARICAMENTO DEI PROGRAMMI

Il COMMODORE 64 accetta programmi da disco, cartuccia o nastro a cassetta. Questo significa che puoi utilizzare software prescritto semplicemente caricandolo. Ma, ancor più importante, il 64 ti permette di salvare i tuoi programmi personali per un riutilizzo. Tutto ciò che devi fare per riutilizzare un programma che hai scritto e salvarlo su disco o nastro, è di caricarlo ed eseguirlo.
Quando col tuo COMMODORE 64 adoperi i nastri o dischi, assicurati che il gestore-disco o l'unità a cassette siano collegati correttamente.

Caricamento Da Cartuccia

Con il tuo 64 puoi utilizzare una serie particolare di programmi e giochi su cartuccia. I programmi comprendono un'ampia varietà di applicazioni commerciali e personali. I giochi sono proprio di tipo arcade, non imitazioni. Per caricare i giochi ed altre cartucce segui questi passi:

1) Spengi il tuo COMMODORE 64. PRIMA DI INSERIRE O RIMUOVERE LE CARTUCCE, DEVI SPENGERE IL TUO COMMODORE 64. SE NON LO FAI, CORRI IL RISCHIO DI DANNEGGIARE LA CARTUCCIA ED IL COMPUTER.
2) Nella fessura sul retro del tuo computer, inserisci la cartuccia con l'etichetta in alto.
3) Accendi il tuo 64.
4) Inizia il gioco immettendo la chiave START che si trova nel foglio delle istruzioni del gioco.

Caricamento Da Nastro a Cassetta Preconfezionato

Puoi anche acquistare software preconfezionato su nastro a cassetta. Queste cassette sono proprio come quelle musicali che puoi riprodurre su uno stereo.

1) Inserisci la cassetta nel tuo registratore 1530 DATASSETTE.
2) Assicurati che il nastro sia completamente avvolto all'inizio del primo lato.

3) Digita LOAD sulla tastiera. Il computer risponderà visualizzando:

PRESS PLAY ON TAPE

4) Premi il tasto PLAY sul tuo DATASSETTE. Lo schermo si vuota fino a quando il computer non trova il programma. Quindi lo schermo visualizza il messaggio

FOUND (NOME PROGRAMMA).

5) Premi il tasto <C=> che caricherà realmente il programma dentro il computer. Se vuoi fermare il caricamento, premi il tasto <RUN/STOP>.

Caricamento dei Propri Programmi Da Nastro a Cassetta

Il COMMODORE 64 ti permette di scrivere e salvare programmi su qualsiasi marca di nastro a cassetta. Tutto quello che ti occorre è un registratore 1530 DATASSETTE e lo stesso tipo di nastro vuoto che utilizzeresti per registrare musica su un riproduttore stereofonico a nastro. Per caricare un programma che hai scritto e salvato su nastro, segui questi semplici passi:

1) Riavvolgi il nastro all'inizio.
2) Digita LOAD "NOME PROGRAMMA". Se non ricordi il nome del programma, basta digitare LOAD. In questo caso verrà caricato in memoria il primo programma trovato sul nastroa.
3) Premi <RETURN>. Il computer risponde con:

PRESS PLAY ON TAPE

4) Premi il tasto PLAY del DATASSETTE. Mentre il computer cerca il programma, la schermata si svuota. Una volta trovato il programma, lo schermo visualizza questo messaggio:

FOUND NOME PROGRAMMA

5) Premi il tasto <C=> per caricare veramente il programma. Durante il caricamento la schermata si risvuota. Una volta finito il caricamento, la schermata ritorna normale e riappare il prompt READY. Se vuoi annullare il caricamento, premi il tasto <RUN/ STOP>.

NOTA: Quando nella memoria del computer carichi un nuovo programma, qualsiasi istruzione e programma esistente non salvato verrà cancellato e perso definitivamente. Per questo motivo, prima di caricare un nuovo programma assicurati di salvare qualsiasi cosa che tu voglia conservare.
Dopo che il tuo programma è stato caricato, puoi eseguirlo (RUN), LISTarlo o farci modifiche. In quest'ultimo caso, però, ricordati di risalvarlo se vuoi conservare la nuova versione.

Caricamento Da Disco

I dischi, che spesso vengono chiamati "dischi flessibili", sono veramente facili da utilizzare. Il vantaggio dei dischi, rispetto ai nastri, è che sui dischi puoi trovare più velocemente i dati che contengono e, inoltre, puoi salvarci molti più dati rispetto ad un nastro.
I passi che seguono sono validi sia per i dischi a caricamento preprogrammato che per i dischi programmati da te.

1) Inserisci un disco dentro il tuo gestore-disco. Assicurati che l'etichetta del disco sia rivolta verso l'alto. Metti il disco in modo che l'estremità etichettata entri per ultimo. Cerca una piccola tacca sul disco (potrebbe essere coperta da un pezzetto di nastro). Quando inserisci il disco, questa tacca deve trovarsi sul lato sinistro, presumendo che tu sia orientato verso il tuo computer. Assicurati che il disco entri per tutto il tragitto.
2) Dopo aver inserito il disco, chiudi la porta di protezione; basta spingere in basso la levetta.
3) Digita LOAD "NOME PROGRAMMA",8. L' 8 è il codice per i dischi e questa volta lo devi digitare per far capire al computer che stai caricando un disco.

NOTA: Puoi CARICARE il primo programma mettendo il segno * al posto del nome programma: LOAD"*",8.

4) Premi il tasto <RETURN>. Il disco comincerà a girare ed il tuo schermo visualizzerà:

```
SEARCHING FOR PROGRAM NAME
LOADING
READY
_
```

5) Quandoo sullo schermo appare READY ed il cursore, digita RUN. Il tuo software è pronto all'uso.

2.3. COME FORMATTARE UN NUOVO DISCO

Quando utilizzi un disco nuovo (vergine) per la prima volta, non programmato, devi formattarlo. La formattazione, anche chiamata intestazione, prepara il disco per fare cose come suddividerlo in blocchi. La formattazione crea anche una cartella che usi come lavagna per i file che salvi sul disco. NON formattare un disco pre-programmato o ne cancellerai tutti i contenuti. Devi formattare SOLO dischi nuovi e non dischi che hanno programmi, a meno che tu non voglia eliminare tutti i suoi contenuti per riutilizzarlo.
Per formattare un disco nuovo, usa questa particolare versione dei comandi OPEN e NEW:

```
OPEN 1,8,15,"N0:<nome>,<id>"
```

N0 dice al computer di formattare (NEW) il disco nel gestore 0. Se, attraverso un'interfaccia adeguata, hai collegato due gestori disco, intesta (formatta) il gestore 0.
Il <nome> usato in questo comando entra nella cartella come nome del disco.
Dagli un nome qualsiasi fino a 16 caratteri.
L'<id> è formato da due caratteri qualsiasi. Dai al disco qualsiasi <id> che vuoi, ma ad ogni disco dovrai dare un codice <id> diverso.
Non appena il gestore-disco spenge la luce, digita CLOSE 1 e premi <RETURN>.

STAI ATTENTO! La formattazione di un disco cancella tutte le informazioni sul disco, se ce ne sono. Formatta solo i dischi nuovi o un disco che desideri cancellare. Qui ci sono alcuni esempi di comandi di formattazione per formattare un disco:

```
OPEN 1,8,15,"N0:MIOFILE,A3"
OPEN 1,8,15,"N0:$RECORDS,02"
```

Adesso che sai preparare un disco, sei pronto ad utilizzare i dischi per scrivere e salvare programmi sul tuo COMMODORE 64. L'Appendice S contiene ulteriori informazioni sul comando OPEN.

2.4. SALVATAGGIO DEI PROGRAMMI

Quando vuoi riutilizzare un programma che hai scritto, assicurati di SALVARLO prima di CARICARE un altro programma. Se non lo fai, perderai il programma. Quando modifichi un programma salvato, devi SALVARLO un'altra volta per tenere la versione nuova.
Quando risalvi un programma, in pratica sostituisci la versione vecchia con quella nuova. Se vuoi mantenere sia la vecchia che la versione modificata, quando salvi quest'ultima gli devi dare un altro nome.

Salvataggio su disco

Quando vuoi SALVARE su disco un programma da te scritto, segui questi semplici passi:

1) Immetti SAVE"NOME PROGRAMMA",8. 8 è il codice per i dischi e dice al computer che stai utilizzando un disco.

2) Premi <RETURN>. Il disco comincia a fare rumore e, non appena salvato il programma, il computer visualizza questo messaggio:

```
SAVING NOME PROGRAMMA
OK
READY
_
```

Salvataggio su Nastro a Cassetta

Quando vuoi SALVARE su nastro un programma che hai scritto, segui questi passi:

1) Immetti SAVE"NOME PROGRAMMA". Il nome del programma che usi può essere lungo fino a 16 caratteri.
2) Premi il tasto <RETURN>. Il computer visualizza il messaggio PRESS RECORD AND PLAY ON TAPE.
3) Premi i due tasti RECORD e PLAY sul tuo registratore DATASSETTE. La schermata si svuota e diventa del colore del bordo. Non appena SALVATO il programma, riappare il prompt READY.

2.5. LISTATO DI UNA CARTELLA DI PROGRAMMI DA DISCO

Quando SALVI i programmi su un disco, il computer crea automaticamente una tabella dei contenuti, o CARTELLA, con i nomi dei programmi sul disco. Tu puoi visualizzare questa cartella per vedere quali programmi si trovano sul tuo disco. Segui questi passi:

1) Immetti: LOAD"$",8 e premi <RETURN>. Il computer visualizza questo messaggio:

```
SEARCHING FOR $
LOADING
READY
_
```

2) Immetti: LIST e premi <RETURN>.

Adesso i nomi dei programmi presenti sul disco sono visualizzati sullo schermo.

3. IL BASIC PER PRINCIPIANTI

3.1. STAMPA E CALCOLO

Se non conosci il BASIC, questo paragrafo ti insegna come eseguire alcune cose semplici come stampare parole e calcolare problemi.
L'istruzione PRINT dice al computer 64 di stampare qualcosa sullo schermo. PRINT è uno dei comandi più utili e potenti nel linguaggio BASIC. Puoi utilizzarlo per visualizzare qualcosa, compresa la grafica ed i risultati dei calcoli. Per utilizzare il comando PRINT segui questi passi:

1) Immetti la parola PRINT. Questo dice al computer che tipo di compito vuoi fargli fare.
2) Immetti una virgoletta. Questo fa sapere al computer dove comincia il messaggio che vuoi stampare.
3) Immetti qualsiasi cosa che tu voglia stampare sullo schermo.
4) Immetti una virgoletta di chiusura. Questa fa sapere al computer dove finisce il messaggio che vuoi stampare.
5) Premi il tasto <RETURN>. Questo dice al computer di eseguire le tue istruzioni che, in questo caso, sono di stampare il tuo messaggio esattamente come lo hai immesso.

Se segui questi passi, il computer stampa il tuo messaggio e visualizza il prompt READY. Potrebbe essere come questo:

PRINT "AMO IL MIO COMMODORE"	Immetti questo e premi <RETURN>
AMO IL MIO COMMODORE	Il computer stampa questo sullo schermo
READY	
_	

Il 64 stampa qualsiasi cosa racchiusa tra virgolette. Ricordati di mettere entrambi le virgolette (di apertura e di chiusura).
Se fai un errore scrivendo tutta l'istruzione, usa il tasto INST/DEL per correggerlo. Prima di premere il tasto <RETURN> puoi modificare tutti i caratteri che vuoi.
Se hai fatto un errore di cui non ti sei accorto prima di premere il tasto <RETURN>, il computer non potrà seguire le tue istruzioni e, anzi, visualizzerà un messaggio d'errore per aiutarti a capire quello che hai fatto di sbagliato. Per esempio:

```
?SYNTAX ERROR
```

Se ricevi questo messaggio, controlla quello che hai immesso per scoprire dove hai fatto un errore. Normalmente si tratta di un errore fatto sul comando - per esempio PRONT invece di PRINT - e non un errore nel messaggio perché, come detto prima, il computer stamperà qualsiasi messaggio fra virgolette, a costo di stampare AMO IL MIO COMODORE invece di AMO IL MIO COMMODORE. Il computer è molto preciso e non può seguire le istruzioni che contengono errori d'ortografia od altri errori. Per evitare errori, assicurati di digitare le cose nella forma giusta.
Ricordati che il modo migliore per conoscere il BASIC ed il tuo 64 è di provare cose diverse e vedere cosa succede.

USO DI PRINT PER CALCOLARE

Puoi utilizzare PRINT per eseguire più che visualizzare quello che inserisci tra virgolette. Puoi utilizzarlo anche per eseguire calcoli e visualizzare automaticamente i risultati. Segui questi passi:

1) Immetti PRINT.
2) Immetti il calcolo che vuoi risolvere ma, questa volta, NON racchiuderlo fra virgolette.
3) Premi il tasto <RETURN>. Il computer visualizza la risposta seguito dal prompt READY.

Ecco un esempio:

PRINT 12 + 12	Inserisci questa riga e premi <RETURN>
24	Il computer visualizza la risposta
READY	
_	

Quando vuoi far risolvere un problema al computer, assicurati di non mettere le virgolette. Se digiti il problema dentro le virgolette, il computer riterrà che tu voglia solo visualizzare il problema e non risolverlo. Per esempio:

PRINT "12 + 12"	Inserisci questa riga e premi <RETURN>
12 + 12	Il computer visualizza quello che trova tra virgolette
READY	
_	

Per cui tutto ciò che devi fare per utilizzare PRINT, come fosse un calcolatore, è di tralasciare le virgolette. Puoi utilizzare PRINT per sommare, sottrarre, moltiplicare e dividere, ma puoi utilizzarlo anche con esponenti ed eseguire funzioni matematiche superiori come rappresentare le radici quadrate.

3.2. FUNZIONI MATEMATICHE

ADDIZIONE

Usa il segno più (+) per dire al computer di sommare numeri. Dopo aver inserito PRINT ed il calcolo, ricordati sempre di premere <RETURN> per dire al computer di eseguire le tue istruzioni.

PRINT 12 + 9	Immetti questo e <RETURN>
21	Il computer visualizza questo

SOTTRAZIONE

Per sottrarre usa il segno meno (-). Alla fine del calcolo premi il tasto <RETURN>. Per esempio:

PRINT 12 - 9	Immetti questo e <RETURN>
3	Il computer visualizza questo

MOLTIPLICAZIONE

Per moltiplicare usa l'asterisco (*). Non puoi utilizzare la x convenzionale perché il computer la riterrà la lettera x e non il segno di moltiplicazione. Premi <RETURN> alla fine del calcolo. Per esempio:

PRINT 12 * 12	Immetti questo e <RETURN>
144	Il computer visualizza questo

DIVISIONE

Per la divisione usi il segno della barra (/). Premi il tasto <RETURN> dopo aver digitato il calcolo. Per esempio:

PRINT 144/12	Immetti questo e <RETURN>
12	Il computer visualizza questo

ESPONENTI

Per elevare un numero alla potenza usa l'accento circonflesso (^). Dopo aver inserito il calcolo premi il tasto <RETURN>. Per esempio, per trovare il 12 alla quinta potenza immetti questo:

PRINT 12^5	Immetti questo e <RETURN>
248832	Il computer visualizza questo

Questo calcolo è lo stesso di:

```
PRINT 12*12*12*12*12
248832
```

> SUGGERIMENTO:
> Il BASIC possiede delle scorciatoie che rendono più veloce la programmazione. Una scorciatoia è l'abbreviazione delle parole chiave del BASIC. Per esempio, al posto di PRINT puoi utilizzare un ?. In ogni parte di questo libro ti mostreremo altre abbreviazioni per le parole chiave del BASIC. L'Appendice D elenca queste abbreviazioni e mostra cosa viene visualizzato sullo schermo quando digiti la forma abbreviata.

3.3. CALCOLI MULTIPLI SU UNA RIGA

L'ultimo esempio mostra che puoi eseguire più di un calcolo sulla stessa riga, ed anche formare il calcolo con operazioni differenti. Per esempio:

? 3 * 5 - 7 + 2	Immetti questo e <RETURN>. Ricorda che ? sta per PRINT
10	Il computer visualizza questo

Fin qui i nostri esempi hanno usato numeri piccoli e problemi semplici. Ma il 64 può eseguire calcoli molto più complessi. L'esempio successivo somma numeri grandi.
Da notare:

1) nella notazione anglosassone, il numero 78956.87 non ha una virgola tra l' 8 ed il 9 per separare le migliaia dalle altre unità. In BASIC non puoi utilizzare le virgole in questo modo. Il BASIC ritiene che le virgole indichino numeri nuovi, per cui riterrebbe il 78,956.87 essere formato da due numeri: il 78 ed il 956.87.
2) nella notazione italiana basta ricordare che la notazione anglosassone agisce esattamente al contrario. In pratica, come nel BASIC, la notazione anglosassone usa il punto per indicare i numeri decimali (in Italia viene messo per separare le migliaia) e la virgola per separare le migliaia (in Italia viene utilizzata per separare i decimali).

Ricordati di premere <RETURN> dopo aver digitato il problema.

```
? 1234.5 + 3457.8 + 78956.87
83649.17
```

Il prossimo esempio usa un numero a dieci cifra. Il 64 può lavorare con numeri che hanno fino a dieci cifre, ma nella soluzione può visualizzarne solo nove arrotondando il risultato. In pratica se l'ultimo decimale è un cinque o superiore verrà arrotondato al numero superiore, se un quattro o inferiore verrà arrotondato al numero inferiore. Questo significa che un risultato come 12123123.45 verrà arrotondato come 12123123.5. A causa dell'arrotondamento il computer non dà la stessa soluzione che otterresti sommando a mano questi numeri. In questo caso la soluzione è 12131364.817. Puoi osservare la differenza arrotondando le strutture.

```
? 12123123.45 + 345.78 + 7895.687
12131364.9
```

Il 64 stampa numeri tra 0,01 e 999.999.999 usando la notazione standard, e lasciando le virgole nei numeri grandi. I numeri che escono da questo campo vengono stampati in notazione scientifica che ti permette di esprimere un numero assai grande o assai piccolo come una potenza del 10. Per esempio:

```
? 123000000000000000
1.23E+17
```

Un altro modo di esprimere questo numero è 1.23* 10 ^ 17. Il 64 usa la notazione scientifica per i numeri con molte cifre per renderli più facilmente leggibili.
Naturalmente c'è un limite ai numeri che può gestire il computer, anche con la notazione scientifica. Questi limiti sono:

Numeri più grandi:	+/- 1.70141183E+38
Numeri più piccoli:	+/- 2.93873588E-39

3.4. ORDINE DI ESECUZIONE NEI CALCOLI

Se hai provato ad eseguire alcuni calcoli misti, sicuramente non avrai ottenuto i risultati aspettati. Questo perché il computer esegue i calcoli in un dato ordine.
In questo calcolo:

20 + 8 / 2

la soluzione è 14 se prima sommi 20 a 8 e poi dividi 28 per 2. Ma la soluzione è 24 se prima dividi 8 per il 2 e poi sommi 20 e 4. Sul 64 otterrai sempre 24 perché il computer esegue sempre i calcoli nell'ordine del secondo esempio. I problemi vengono risolti da sinistra a destra, ma all'interno di quel meccanismo generico alcuni tipi di calcolo hanno la precedenza sugli altri. Ecco l'ordine di precedenza:

Primo :	-	il segno meno per i numeri negativi, non per la sottrazione.
Secondo:	^	l'esponente, da sinistra a destra
Terzo :	* /	moltiplicazione e divisione, da sinistra a destra
Quarto:	+ -	addizione e sottrazione, da sinistra a destra

Questo significa che il computer, prima di fare qualsiasi cosa, esamina tutto il calcolo alla ricerca di numeri negativi. Quindi cerca gli esponenti; esegue tutte le moltiplicazioni e le divisioni, ed infine somma e sottrae.

Questo spiega perché 20 + 8 / 2 fa 24: l'8 viene diviso per 2 prima che venga sommato al 20 perché la divisione ha la precedenza sull'addizione. C'è un modo facile per ignorare l'ordine di precedenza: racchiudere fra parentesi ogni calcolo che vuoi far risolvere per primo. Se aggiungi le parentesi all'equazione mostrata sopra, ecco cosa succede:

```
? (20 + 8) / 2
14
```

Ottieni 14 perché le parentesi permettono al 20 ed all' 8 di essere sommati prima che avvenga la divisione.

Qui ci sono altri esempi che mostrano in che modo cambiare l'ordine, e la soluzione, mettendo le parentesi:

```
? 30 + 15 * 2 - 3
57

? (30 + 15) * 2 - 3
87

? 30 + 15 * (2 - 3)
15

? (30 + 15) * (2 - 3)
-45
```

L'ultimo esempio ha due calcoli fra parentesi. Come al solito, vengono valutati da sinistro a destra per poi risolvere il resto del problema. Quando hai più di un calcolo fra parentesi, puoi controllare ulteriormente l'ordine utilizzando parentesi dentro parentesi. Il problema tra le parentesi più interne viene risolto per primo. Per esempio:

```
? 30 + (15 * (2 - 3))
15
```

In questo caso il 3 viene sottratto da 2, poi il 15 viene moltiplicato per -1, ed il risultato (-15) viene sommato al 30. Via via che provi a risolvere i calcoli, acquisirai familiarità con l'ordine in cui vengono risolti i calcoli misti.

3.5. COMBINAZIONI POSSIBILI DEL PRINT

I computer 64 ti permettono di combinare i due tipi di istruzioni print che hai letto in questo libro. Ricordati che qualsiasi cosa racchiusa fra le virgolette viene visualizzata esattamente come viene digitata. L'esempio successivo mostra come puoi combinare i vari tipi dell'istruzione PRINT. L'equazione racchiuso fra virgolette viene visualizzato senza essere risolta. L'equazione senza

virgolette viene risolto. Il punto e virgola separa le due parti dell'istruzione PRINT (il punto e virgola si intende senza spazi).

```
? "5 * 9 ="; 5 * 9          Immetti questo e <RETURN>
5 * 9 = 45                  Il computer visualizza questo
```

Ricorda, solo la seconda parte della frase verrà calcolata. Le due parti sono separate da un punto e virgola. Le parti di un'istruzione mista del PRINT devono sempre essere separare con qualche tipo di punteggiatura affinché lavori nel modo voluto. Se usi una virgola al posto del punto e virgola, al momento della visualizzazione tra le due parti ci sarà maggior spazio. Il punto e virgola unisce le parti senza spazi.
La schermata del 64 è organizzata in 4 zone di 10 colonne ciascuna. Quando usi una virgola per separare le parti di un'istruzione PRINT, la virgola opera come una tabulazione mettendo ciascun risultato nella zona successiva. Per esempio:

```
? "TOTALE:";95,"SCARSITÀ:";15
TOTALE: 95          SCARSITÀ: 15
```

Se hai più di quattro risultati, verranno automaticamente visualizzati sulla riga successiva. Per esempio:

```
? 2 * 3,4 - 6,2 ^ 3,6 / 4,100 + (-48)
6          -2          8          1.5
52
```

Ecco la differenza quando metti il punto e virgola:

```
? 2 * 3;4 - 6;2 ^ 3;6 / 4;100 + (-48)
6 -2  8  1.5  52
```

Nella formattazione dell'istruzione PRINT puoi utilizzare la differenza che c'è tra la virgola ed il punto e virgola per creare visualizzazioni complesse.

4. SCRITTURA DI SEMPLICi PROGRAMMi BASIC

Fin qui questo libro ti ha mostrato in che modo effettuare cose semplici con il tuo 64. Hai provato ad immettere singole righe di istruzioni nel tuo computer ed ottenere risultati immediati premendo il tasto <RETURN>. Questo modo facile di fare le cose sul tuo computer viene chiamata modalità IMMEDIATA (o DIRETTA) o modalità CALCOLATORE.
Probabilmente, però, vorrai utilizzare il tuo computer per compiti più complessi, che utilizzino più di una istruzione. Quando in un PROGRAMMA unisci una quantità di istruzioni, avrai a disposizione tutta la potenzialità del tuo 64.
Per vedere quanto sia facile scrivere il tuo primo programma sul 64, segui questi passi:

1) Vuota lo schermo tenendo premuto il tasto <SHIFT> mentre premi il tasto <CLR/HOME>.
2) Immetti NEW e premi <RETURN>. Questo comando elimina le informazioni che potrebbero trovarsi ancora nella memoria del computer dopo le tue prove.
3) Immetti le due righe di seguito, esattamente come si presentano:

```
10 ? "COMMODORE 64"
20 GOTO 10
```

4) Ricorda di premere il tasto <RETURN> dopo aver scritto ciascuna riga. Qui potrai notare qualcosa di diverso: dopo aver immesso la prima riga e premuto <RETURN>, il computer non risponderà al comando PRINT come invece ha fatto quando hai dato lo stesso tipo di comando nelle prove precedenti. La differenza sta nel fatto che adesso il comando inizia con un numero di

riga (10). Quando usi un numero di riga, il computer sa che stai scrivendo un programma e, prima di seguire ognuna delle tue istruzioni, aspetterà che tu finisca tutto il programma.

5) Immetti RUN e premi <RETURN>. Il comando RUN fa sapere al computer che hai terminato l'introduzione delle istruzioni del programma e sei pronto a fargli eseguire le tue istruzioni. Ecco cosa succede quando esegui (RUN) questo programma:

```
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
…
```

6) Arresta l'esecuzione del programma premendo il tasto <STOP>. Il computer continua a seguire i tuoi ordini, stampando COMMODORE 64 di continuo, fino a quando non lo interrompi con il tasto <STOP>. Ecco cosa appare sullo schermo quando premi STOP.

```
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
BREAK IN 10
READY
```

Questo semplice programma introduce diversi concetti importanti che sono alla base di tutta la programmazione.

4.1. NUMERI DI RIGA

Nel paragrafo precedente abbiamo accennato al fatto che i numeri di riga dicono al computer che stai scrivendo un programma, ma gli dicono anche in quale ordine vuoi fargli eseguire le istruzioni del tuo programma. Senza i numeri di riga che dicono al computer quali istruzioni seguire, il computer non saprebbe da quale cominciare.
Più lungo e più complesso sarà il tuo programma, più importante sarà ricordarsi che il computer conta sul fatto di dirgli ***QUANDO*** eseguire le cose, ma anche ***QUELLO*** che deve fare. Una bella cosa a questo proposito è che puoi immettere la riga 20 prima della riga 10 ed il controlla eseguirà il programma nel giusto ordine mettendo la riga 20 DOPO la riga 10. Il computer non controlla l'ordine delle righe visualizzate sullo schermo ma le eseguirà dal numero più piccolo al più grande.
Un altro vantaggio dei numeri di riga è che puoi utilizzare il numero come riferimento all'istruzione sulla riga. Quando vuoi ritornare e ripetere l'esecuzione di un'istruzione, tutto quello che devi fare è riferirti ad essa attraverso il numero di riga con un'istruzione GOTO, come hai visto nell'esempio sopra. In pratica il numero di riga diventa un’etichetta alla quale fare riferimento.

4.2. L'ISTRUZIONE GOTO

Quando hai detto al computer di ESEGUIRE il programma dell'esempio sopra, la frase COMMODORE 64 è stata stampata ripetutamente, invece che una volta, a causa dell'istruzione GOTO nella riga 20. L'istruzione GOTO dice al computer di andare direttamente su una specifica riga del programma (che deve esistere) in cui il computer, prima di proseguire, dovrà eseguirne le istruzioni. Puoi utilizzare un'istruzione GOTO per dire al computer di ritornare su una riga che sia già stata eseguita, oppure saltare in avanti, anche se questo significa non eseguire alcune righe del programma.

In pratica, nel nostro esempio, succede questo: il computer esegue l'istruzione della prima riga (la 10) e stampa sullo schermo COMMODORE 64. Poiché su questa riga non ci sono altre istruzioni, il computer passa alla riga seguente (la 20) ed esegue l'istruzione che trova qui. L'istruzione GOTO dice al computer di andare direttamente alla riga indicata (in questo caso ancora la 10). Qui il computer riceve nuovamente l'ordine di stampare COMMODORE 64, lo esegue e ripassa nuovamente alla riga successiva (la 20) che la riporta alla riga 10... in un ciclo senza fine.
Questo ciclo, o ripetizione, viene chiamato LOOP. Poiché l'esempio non dà al computer un'uscita dalla sequenza, il ciclo si ripete senza fine e, per fermare il programma, devi utilizzare il tasto <STOP>. È sempre conveniente aggiungere un'istruzione che termini la sequenza in modo da non dover utilizzare il tasto <STOP>. Più avanti, in questo capitolo, spiegheremo meglio come fermare le sequenze.

4.3. USO DEL COMANDO LIST

Adesso che hai interrotto l'esecuzione del programma d'esempio, immetti LIST e premi <RETURN>. Il tuo programma viene visualizzato integralmente, perché si trova ancora nella memoria del computer, anche se ne hai interrotto l'esecuzione. L'unica differenza è che il computer ha sostituito il tuo ? con la parola PRINT. Questo non ha effetto sul tuo programma, è proprio il modo in cui il computer fa le cose (ricorda che il ?, come altre scorciatoie, serve solo a te per velocizzare l'immissione delle istruzioni e corrisponde al comando PRINT, cosa che giustamente riporta il computer quando LISTi il programma). Un'altra prerogativa del comando LIST è che il computer visualizza le righe del programma nel giusto ordine crescente, anche se tu hai immesso righe fuori da questo ordine (per esempio hai inserito la riga 20 prima della riga 10).
Una delle differenze principali tra lo scrivere programmi ed inserire righe singole in modalità immediata/calcolatore, è quella che, una volta eseguita e vuotato lo schermo, l'istruzione immediata viene persa permanentemente, mentre il programma, fino a quando non ne inizi uno nuovo, puoi sempre riportarlo sullo schermo con il comando LIST e, da qui, poterlo modificare, salvare o eseguire un'altra volta.

4.4. SUGGERIMENTI PER L'EDITAZIONE

Se fai un errore su una riga che hai immesso, o se vuoi modificare una riga, il 64 ti offre una quantità di possibilità per l'editazione.

1) Puoi re-immettere una riga quando vuoi ed il computer sostituirà automaticamente la nuova riga con quella vecchia. Tutto ciò che devi fare per sostituire una riga, è di utilizzare lo stesso numero di riga. Per esempio:

    a. 10 ? "IL MIO NOME E' SARAH"
    b. 20 ? "SONO NATA IN CALIFORNIA"
    c. 20 ? "VIVO IN PENNSYLVANIA"
    d. RUN
    e. IL MIO NOME E' SARAH
    f. VIVO IN PENNSYLVANIA

   Come puoi vedere, la prima riga 20 non verrà mai eseguita perché sostituita dalla seconda riga. Se adesso immetti il comando LIST, vedrai che nel programma ci sarà solo la seconda riga 20.
2) Puoi facilmente eliminare una riga che non vuoi più, immettendo solo quel numero di riga e premendo il tasto <RETURN>. Se adesso immetti LIST, vedrai che la riga non c'è più, come pure il numero di riga.
3) Puoi facilmente editare una riga esistente. Usa i tasti CuRSoRe per riportare il cursore sulla riga che vuoi modificare, ed edita la riga nel modo che più ti aggrada. Non appena premi il tasto <RETURN>, la riga editata sostituirà la riga vecchia. Ricordati di utilizzare il tasto <INST/DEL> per inserire o cancellare i caratteri.

Non appena finisci l'editazione, puoi controllare di nuovo il tuo programma, per verificarne le modifiche, immettendo il comando LIST. Ricorda che LIST mette anche le righe in ordine numerico, nel caso in cui tu le abbia messe disordinate.
Prova ad editare il nostro primo programma d'esempio aggiungendo un punto e virgola alla fine della riga e togliendo il 64. Dopo aver finito le modifiche, assicurati di portare il cursore oltre la riga 20 prima di avviare (RUN) il programma. Ecco come si presenta adesso il programma:

```
LIST
10 PRINT "COMMODORE";
20 GOTO 10

RUN
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
BREAK IN 10
READY
```

4.5. COME UTILIZZARE LE VARIABILI

Una variabile è un simbolo che indica un valore. Qualche volta, prima che tu avvii un programma, il valore di una variabile è sconosciuto. Uno degli scopi di un programma può essere di trovare uno o più valori per una variabile. Esamina questa riga di programma:

```
20 LET X = 28 + Y
```

Il segno = significa "diventa" o "prendi il valore di". L'istruzione LET è facoltativa e può essere omessa.

In questa equazione X ed Y sono variabili. Supponi che X indichi il numero dei giorni in un mese. Una delle cose migliori di una variabile è che puoi riutilizzarla in un programma, per cui X può indicare i giorni di tutti i mesi e non solo di uno. Qui entra in ballo la Y. Poiché tutti i mesi hanno perlomeno 28 giorni, Y indica i giorni sopra il 28. Più avanti, in questo capitolo, c'è un programma che dà i valori a queste due variabili.
Adesso la cosa più importante è di capire come lavorano le variabili perché con il tuo computer ti permettono di eseguire compiti complessi. Le variabili ti permettono anche di scrivere programmi da riutilizzare in altre parti. Immagina che il tuo computer contenga un gruppo di piccole fessure, come quelle delle cassette per la posta. Quando scrivi un programma, puoi utilizzare alcune di queste fessure in cui custodire i valori. Tutto quello che devi fare è di dare un nome alle fessure di cui hai bisogno e, durante il programma, puoi inserire valori in ciascuna fessura richiamandone il nome. Per esempio, nell'equazione sopra descritta, noi abbiamo utilizzato due fessure chiamandole una X ed una Y. All'inizio di un programma queste fessure hanno i nomi ma sono vuote. Ecco cosa succede quando metti un valore nella fessura Y:

```
+--------+--------+--------+--------+--------+--------+--------+
|   X    |   Y    |        |        |        |        |        |
+--------+--------+--------+--------+--------+--------+--------+
|        |   3    |        |        |        |        |        |
+--------+--------+--------+--------+--------+--------+--------+
```

Adesso la variabile Y ha il valore 3. Puoi dare ad Y questo valore scrivendo questa semplice istruzione:

```
10 Y = 3
```

Poiché X equivale a 28 più Y, quando avvii il programma anche la fessura X riceverà una valore.

```
+--------+--------+--------+--------+--------+--------+--------+
|   X    |   Y    |        |        |        |        |        |
+--------+--------+--------+--------+--------+--------+--------+
|   31   |   3    |        |        |        |        |        |
+--------+--------+--------+--------+--------+--------+--------+
```

Ecco come si presenta il programma:

```
10 Y = 3
20 X = 28 + Y
30 ? "IL NUMERO DEI GIORNI IN MAGGIO E' ";X

RUN
IL NUMERO DEI GIORNI IN MAGGIO E' 31
```

Qui c'è un altro programma che usa le variabili:

```
10 X% = 15
20 X = 23.5
30 X$ = "TOTAL:"
40 Y = X% + X
50 ? X$;Y
```

Quando ESEGUI il programma, dopo l'esecuzione della riga 30 le fessure immaginarie sono simili a queste:

```
+--------+--------+--------+--------+--------+
|   X%   |   X    |   X$   |   Y    |        |
+--------+--------+--------+--------+--------+
|   15   |  23.5  | TOTAL: |        |        |
+--------+--------+--------+--------+--------+
```

Al completamento del programma Y ha il valore: 38.5

L'esempio sopra utilizza i tre tipi di varibile:

TIPO	SIMBOLO	DESCRIZIONE	ESEMPI	VALORI D'ESEMPIO
Intero	%	Tutti i numeri	X%, A1%	15, 102, 3
Stringa di testo	$	I caratteri fra virgolette	X$, AB$	"TOTAL:" "DAY 1"
Virgola mobile		Numeri reali (decimali) o tutti i numeri	X, AB	23.5, 12, 1.3E+2

Assicurati di utilizzare le giuste variabili da immettere nei tuoi programmi. Se tenti di fare qualcosa come assegnare una stringa di testo ad una variabile intera, il tuo programma non funzionerà.
Quando assegni i nomi delle variabili ci sono altre cose da ricordare:

- ✓ Un nome di variabile può avere uno o due caratteri, senza contare il simbolo speciale usato con le variabili intere e stringhe di testo.
- ✓ Come nome di una variabile puoi utilizzare più di due caratteri alfabetici, ma il computer riconoscerà solo i primi due. Per questo motivo il computer penserà che PA, PARTNO e PAGENO sia la stessa variabile riferita alla stessa "fessura".
- ✓ Quando per le variabili utilizzi nomi più lunghi, per altre persone risulta più facile leggere un programma, ma quando in un nome usi più di due caratteri assicurati che i primi due siano unici.
- ✓ In un programma puoi utilizzare nomi di varibile X, X% e X$ perché i simboli speciali % e $ rendono unici detti nomi. La stessa cosa vale per A2, A2% e A2$.
- ✓ Il primo carattere di un nome deve essere alfabetico (da A a Z). Il secondo caratteri e gli altri che seguono possono essere sia alfabetici che numerici (da 0 a 9). Ricorda che il computer ignora ogni carattere dopo il secondo, a meno che la terza posizione non sia occupata da un % o $.

- ✓ I nomi delle variabili non possono contiene parole-chiave BASIC, anche chiamate parole riservate. Queste sono le parole come PRINT e RUN che fanno parte del linguaggio BASIC. L'Appendice D elenca tutte le parole riservate del BASIC.

Qui c'è un altro programma d'esempio che ti mostra come usare le variabili. Questo esempio usa anche alcune delle altre cose che hai imparato fin qui.

```
NEW
10 X = 1.05
20 Y = 300
30 Z = X * Y
40 PRINT "POSTI A SEDERE DISPONIBILI:";Y
50 PRINT "BIGLIETTI DISPONIBILI:";Z
60 Y = Y + 1
70 PRINT "PUNTO DI SOVRAPRENOTAZIONE:";Y

RUN
POSTI A SEDERE DISPONIBILI: 300
BIGLIETTI DISPONIBILI: 315
PUNTO DI SOVRAPRENOTAZIONE: 301
```

Le righe (10-30) assegnano i nomi ed i valori alle variabili.
Le righe 40 e 50 stampano un messaggio insieme all'attuale valore delle variabili Y e Z. Da notare che sulla riga 40 il valore di Y è 300.
La riga 60 dà un nuovo valore ad Y, nuovo valore che viene stampato nella riga 70. La riga 60 mostra che in un programma una variabile può avere più di un valore e, inoltre, mostra un'altra delle potenti caratteristiche delle variabili: puoi creare una variabile uguale a se stessa e con un altro valore.
Nell'algebra regolare questo non è permesso, ma nella programmazione questo tipo di istruzione viene usato comunemente. In pratica significa: prendere l'attuale valore di una variabile, unirlo ad un altro valore e sostituire il primo valore della variabile con questo valore nuovo. Puoi anche utilizzare istruzioni come queste:

```
Y = Y - 1
Y = Y + X
Y = Y / 2
Y = Y * (X + 2)
```

4.6. USO DEI CICLI FOR/NEXT

Durante la spiegazione dell'istruzione GOTO in questo capitolo, abbiamo accennato ai cicli. Come ricorderai, i cicli sono esecuzioni ripetute di una o più righe di un programma.
L'istruzione FOR/NEXT ti permette di creare cicli molto utili che controllano le volte che dovrà essere eseguita una parte del programma. L'istruzione FOR imposta un limite alla quantità di volte, che dovrà essere ripetuto in un ciclo, assegnando un campo di valori ad una variabile. Per esempio:

```
FOR COUNT=1 TO 4
```

L'istruzione NEXT designa la fine di un ciclo FOR/NEXT. Quando il programma raggiunge una istruzione NEXT, il computer controlla l'istruzione FOR per vedere se è stato raggiunto il limite del ciclo. Se il limite non è stato raggiunto, il ciclo viene ripetuto e la variabile dell'istruzione FOR viene incrementata di uno. Per esempio, se aggiungi un ciclo FOR/NEXT al programma iniziale di questo capitolo, ecco cosa succede:

```
10 FOR CT =1 TO 4
20 ? "COMMODORE 64"
30 NEXT CT
```

```
RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
READY
_
```

Adesso che hai aggiunto il ciclo FOR/NEXT, non hai più bisogno di interrompere l'esecuzione del programma con il tasto STOP. Il ciclo FOR/NEXT lavora in questo modo:

- ✓ La riga 10 dà alla variabile CT un campo di valori da 1 a 4 e dice al computer di eseguire le righe successive fino a quando CT non abbia raggiunto il valore massimo dato (cioè 4).
- ✓ La riga 20 dice al computer di stampare COMMODORE 64.
- ✓ La riga 30 dice al computer di aggiungere 1 al valore attuale di CT. Fino a quando il valore di CT rimane entro il campo di 1 a 4, il programma ripete la stampa di COMMODORE 64. Quando CT, a forza di aggiungere 1, sarà uguale a 4, la riga verrà eseguita per l'ultima volta. Nel momento in cui la riga 30 aggiungerà ancora 1 a CT, il computer si accorgerà che CT adesso risulta fuori dal campo impostato e, in questo caso, non rieseguirà il ciclo uscendo dal programma (in altri casi, il computer uscirà ugualmente dal ciclo ma con tutta probabilità eseguirà il resto di un programma più complesso di questo).

Per essere sicuri che tu capisca in che modo lavora il ciclo FOR/NEXT, aggiungeremo un'altra istruzione PRINT alla riga 20 che ti permetta di tenere traccia del valore di CT.

```
20 PRINT"COMMODORE 64  ";"COUNT =";CT

RUN
COMMODORE  64  COUNT = 1
COMMODORE  64  COUNT = 2
COMMODORE  64  COUNT = 3
COMMODORE  64  COUNT = 4
```

Come puoi vedere, il programma termina automaticamente non appena CT va fuori dal campo configurato nell'istruzione FOR.
In una istruzione FOR/NEXT puoi incrementare il valore della variabile con valori diversi da 1. Tutto quello che devi fare è aggiunge o la parola STEP o il valore che vuoi utilizzare alla fine dell'istruzione FOR. Per esempio:

```
NEW
10 FOR NB=1 TO 10 STEP .5
20 PRINT NB,
30 NEXT NB
```

La virgola dice al computer di stampare ogni valore a partire dalla prima posizione della successiva zona di 10 spazi.

```
RUN
 1         1.5       2          2.5
 3         3.5       4          4.5
 5         5.5       6          6.5
 7         7.5       8          8.5
 9         9.5       10

NEW
10 FOR A=2 TO 8 STEP 2
20 PRINT A,
30 NEXT A

RUN
2          4         6          8
```

Puoi utilizzare il ciclo FOR/NEXT anche per contare all'incontrario. Quando fai così, assicurati che il tuo STEP sia negativo. Per esempio, se sostituisci la riga 10 con questa:

```
10 FOR A=8 TO 2 STEP -2
```

Ecco come apparirà l'uscita:

```
RUN
8           6           4           2
```

4.7. USO DELL' ISTRUZIONE IF/THEN PER CONTROLLARE I PROGRAMMI

Una istruzione IF/THEN è un altro modo per controllare l'esecuzione del programma. Questa istruzione dice al computer di controllare se (IF) una condizione è vera. In caso positivo, dovrà eseguire le istruzioni che seguono la parola THEN, in caso negativo il programma dovrà proseguire alla riga successiva senza eseguire le istruzioni che si trovano prima di THEN. Per esempio:

```
10 X = 60
20 X = X + 1
30 IF X = 64 THEN PRINT "OTTIENI": END
40 GOTO 20
```

Puoi utilizzare una istruzione IF per iniziare un ciclo o decidere se dovranno essere eseguite certe parti del programma. Per esempio:

```
10 A = 0
20 IF A <= 8 THEN 40
30 END
40 ? "VITE DI FRODO ";A
50 A = A + 2
60 GOTO 20

RUN
VITE DI FRODO  0
VITE DI FRODO  2
VITE DI FRODO  4
VITE DI FRODO  6
VITE DI FRODO  8
```

In questo esempio l'istruzione IF/THEN, alla riga 20, dice al computer di controllare l'attuale valore di A. Se (IF) A è uguale o minore di 8, allora (THEN) il programma salta alla riga 40 e continua eseguendo, prima la riga 50 in cui viene incrementato il valore di A, e poi la riga 60 che lo riporta alla riga 20. Se (IF) A è maggiore di 8 o, in altre parole, se (IF) la condizione della riga 20 è falsa, il computer ignora le istruzioni che seguono l'istruzione THEN.
Se (IF) la riga 20 è falsa, allora (THEN) viene eseguita la riga 30. La riga 40 stampa il messaggio ed il valore attuale della variabile.
La riga 50 aggiunge 2 al valore di A ogni volta che viene eseguito il ciclo. Non appena A diventa 10, la riga 20 diventa falsa, la riga 30 diventa vera, ed il programma termina l'esecuzione.
Con l'istruzione IF/THEN puoi utilizzare uno qualsiasi di questi operatori relazionali:

SIMBOLO	SIGNIFICATO
<	Minore di
>	Maggiore di
=	Uguale a
< >	Diverso da
> =	Maggiore o uguale a
< =	Minore o uguale a

5. BASIC AVANZATO

I prossimi pochi capitoli servono a coloro che hanno familiarità con il linguaggio di programmazione BASIC ed i concetti necessari per scrivere programmi avanzati.
Coloro che stanno cominciando ad imparare la programmazione possono ritenere troppo tecniche alcune di queste informazioni per capirle completamente. Nonostante ciò, in due capitoli (FIGURE GRAFICHE e CREAZIONE SUONO) potranno trovare qualche semplice esempio scritto per nuovi utenti. Questi esempi vi daranno un'idea di come utilizzare la grafica sofisticata e le capacità audio del vostro 64. Se volete imparare maggiormente sulla scrittura di programmi in BASIC, esaminate la bibliografia all'Appendice N. Se avete già familiarità con la programmazione BASIC, i capitoli successivi vi aiuteranno iniziando con tecniche avanzate sulla programmazione BASIC. Sulla COMMODORE 64 PROGRAMMER'S REFERENCE GUIDE, disponibile dal vostro commerciante di Commodore 64, troverete ampie informazioni sulla programmazione avanzata.

5.1. SEMPLICE ANIMAZIONE

Puoi utilizzare alcune delle capacità grafiche del 64 mettendo insieme quello che hai imparato fin qui in questo manuale, ed alcuni concetti nuovi. Cerca di entrare nel programma che segue per vedere quello che puoi fare con la grafica. Da notare che dentro un'istruzione PRINT puoi mettere i controlli per il cursore ed i comandi di schermo. Quando in un listato di programma vedi qualcosa come (CRSR a sinistra), tiene premuto il tasto <SHIFT> e premi il tasto <CRSR-LEFT/RIGHT>. Lo schermo rappresenterà la grafica di un cursore a sinistra: due barre verticali invertite. La rappresentazione grafica del tasto <CLR/HOME> con lo <SHIFT> è un cuore invertito (Nota che il termine "invertito" signica il segno grafico bianco su fondo nero invece che segno nero su fondo bianco come normale).

```
NEW
10  REM PALLA RIMBALZANTE
20  PRINT "{CLR/HOME

25  FOR X=1 to 10: PRINT "{CRSR DOWN}": NEXT
30  FOR BL=1 to 40
40  PRINT" O{CRSR LEFT}";: REM (O è uno SHIFT-Q)

50  FOR TM=1 TO 5
60  NEXT TM
70  NEXT BL
75  REM MOVE BALL RIGHT TO LEFT
80  FOR BL=40 TO 1 STEP -1

90  PRINT" {CRSR LEFT}{CRSR LEFT}O{CRSR LEFT}";
100 FOR TM=1 TO 5
110 NEXT TM
120 NEXT BL
130 GOTO 20
```

Separa una istruzione dall'altra

Questi spazi sono intenzionali

SUGGERIMENTO:
Tutte le parole in questo testo verranno compilate su una riga. In questo caso, fino a quando non premi <RETURN>, il tuo 64 si porterà automaticamente sulla riga successiva anche se si trova nel mezzo di una istruzione.

Non appena esegui (RUN) questo programma, si visualizzerà una palla che rimbalza e che si muove attraverso la schermata, da sinistra a destra e ritorno. Esamina attentamente il programma per vedere come viene fatto.

```
              NEW
              10  REM PALLA RIMBALZANTE
  +---------> 20  PRINT "{CLR/HOME}"
  |  +------> 25  FOR X=1 to 10: PRINT "{CRSR DOWN}": NEXT
  |  |        30  FOR BL=1 to 40
  |  |        40  PRINT" O{CRSR LEFT}";: REM (O è uno SHIFT-Q)
  |  |  +---> 50  FOR TM=1 TO 5
  |  |  +---- 60  NEXT TM
  |  + ------ 70  NEXT BL
  |           75  REM MOVE BALL RIGHT TO LEFT
  |  + -----> 80  FOR BL=40 TO 1 STEP -1
  |  |        90  PRINT" {CRSR LEFT}{CRSR LEFT}O{CRSR LEFT}";
  |  |  +---> 100 FOR TM=1 TO 5
  |  |  +---- 110 NEXT TM
  |  +------- 120 NEXT BL
  +---------- 130 GOTO 20
```

La riga 10 è un commento che ti dice cosa fa il programma. Una istruzione REM non ha effetto sul programma dal momento che non viene considerata come comando eseguibile.
La riga 20 pulisce lo schermo.
La riga 25 stampa 10 volte il comando cursore-giù che serve soltanto a mettere in posizione la palla al centro dello schermo. Senza questa riga la palla si sposterebbe sulla riga in alto dello schermo.
La riga 30 configura un ciclo per muovere la palla su 40 colonne, da sinistra a destra. La riga 40 fa queste tre cose:

2. Stampa uno spazio per cancellare la precedente posizione della palla.
3. Stampa la palla.
4. Esegue un cursore-sinistro per trovarsi in posizione per la cancellazione dell'attuale posizione della palla.

Le righe 50 e 60 configurano un ciclo che rallenta il movimento della palla. Senza questo ciclo la palla si sposterebbe troppo velocemente perché la si possa vedere chiaramente. La riga 70 completa il ciclo configurato nella riga 30 per stampare le palle sullo schermo. Ogni volta che viene eseguito il ciclo, la palla viene mossa a destra di uno spazio. Come puoi vedere dal listato, il programma contiene un ciclo all'interno di un ciclo. In effetti in un ciclo è possibile includere fino a dieci cicli. L'unica volta che puoi avere difficoltà è quando i cicli attraversano tutti gli altri. I cicli devono essere NIDIFICATI all'interno di ogni altro. In altre parole: se inizi il ciclo A e poi inizi il ciclo B all'interno del ciclo A, devi prima finire il ciclo B (il ciclo interno). In questo modo è possibile nidificare fino a nove cicli. Quando scrivi un programma con i cicli, è buona norma disegnare le frecce che indicano l'inizio e la fine del cicli. Se i tuoi cicli vengono attraversati, il computer non potrà capire quello che volevi fargi fare e, di conseguenza, non potrà eseguire il tuo programma.
Le righe dalla 80 alla 120 invertono i passi della prima parte del programma e muovono la palla da destra a sinistra. La riga 90 è leggermente diversa dalla riga 40 perché la palla si sta muovendo nella direzione opposta e devi cancellarla a destra e spostarla verso sinistra.
La riga 130 riporta il programma alla riga 20 per ricominciare tutto il procedimento.

Per fare una variazione al programma, modifica la riga 40 in questo modo:

```
40 PRINT "(SHIFT-Q)"
```

Esegui il programma e guarda adesso cosa succede. Poiché hai lasciato il controllo del cursore, ogni palla disegnata rimane sullo schermo fino a quando non viene cancellata dal movimento della palla da destra a sinistra, cosa che avviene nella seconda parte del programma.

5.2. INPUT

Fino ad ora tutto il programma è stato configurato prima di eseguirlo e, una volta in esecuzione, non era più possibile modificare od aggiungere niente.
L'istruzione INPUT ti permette di inserire informazioni ad un programma ***MENTRE*** è in esecuzione. Non solo il programma deve avere queste informazioni che fornisci, ma non andrà avanti fino a quando non le avrà ricevute. Per avere un'idea di come funzioni INPUT, digita NEW, premi <RETURN> ed inserisci questo programmino.

```
10 INPUT A$
20 PRINT "HAI DIGITATO ";A$
30 PRINT
40 IF A$ = "STOP" THEN END
50 GOTO 10
RUN
? GO
HAI DIGITATO GO
? CONTINUE
HAI DIGITATO CONTINUE
? STOP
HAI DIGITATO STOP
```

DiGITI

Ecco cosa succede in questo programma:

La riga 10 dice al computer di visualizzare un punto interrogativo, con il quale richiedere di INSERIRE un valore per A$, e di aspettare fino a quando non gli fornisci il valore prima di continuare l'esecuzione del programma.
La riga 20 stampa un messaggio ed il valore di A$; la riga 30 stampa una riga vuota di separazione.
La riga 40 dice al computer di terminare il programma se (IF) il valore che inserisci in A$ è STOP.
Se la riga 40 è falsa, il programma passa alla riga 50 che lo fa ritornare alla riga 10 in cui puoi immettere un altro valore. Se invece la riga 40 è vera perché l'ultimo valore inserito per A$ è stato STOP, la riga 50 non verrà eseguita ed il programma si arresta.
Puoi INSERIRE variabili numeriche o di stringa, ed all'istruzione INPUT puoi far stampare un messaggio con un punto interrogativo per descrivere il tipo di INPUT con il quale il computer rimane in attesa. Per esempio, ecco cosa succede quando aggiungi un messaggio d'avviso alla riga 10 dell'esempio precedente:

```
10 INPUT "COSA SCRIVI";A$
```

Il messaggio non può avere più di 38 caratteri

```
RUN
COSA SCRIVI? GO
HAI DIGITATO GO

COSA SCRIVI? STOP
HAI DIGITATO STOP
```

Ecco un esempio più complesso che descrive un bel po' di quello che è stato presentato fin qui, inclusa l'istruzione INPUT.

```
NEW
 1 REM PROGRAMMA DI CONVERSIONE DELLA TEMPERATURA
 5 PRINT "(SHIFT-CLR/HOME)"
10 PRINT "CONVERTO DA FAHRENHEIT O CELSIUS (F/C): ": INPUT A$
20 IF A$ = "" THEN 10
30 IF A$ = "F" THEN 100
40 IF A$ <> "C" THEN END
50 INPUT "INSERISCI I GRADI CELSIUS: ";C
60 F = (C*9)/5+32
70 PRINT C;" GRADI CELSIUS ="; F ;"GRADI FAHRENHEIT"
80 PRINT
```

```
90 GOTO 10
100 INPUT "INSERISCI I GRADI FAHRENHEIT: ";F
110 C = (F-32)*5/9
120 PRINT F;" GRADI FAHRENHEIT ="; C ;"GRADI CELSIUS"
130 PRINT
140 GOTO 10
```

La riga 10 usa l'istruzione INPUT per stampare un messaggio ed aspettare che tu immetta un valore per A$.
Le righe 20, 30 e 40 controllano quello che hai immesso e, in base alla risposta, dicono al computer dove andare. La riga 20 dice al computer di ritornare alla riga 10 e richiedere l'INPUT se (IF) non fosse stato immesso niente (in pratica se è stato premuto <RETURN> senza altri valori). La riga 30 dice al computer di saltare alla rigae 100 ed eseguire la conversione da Fahrenheit a Celsius se (IF) il valore digitato per A$ era F.
La riga 40 si assicura che tu non abbia immesso qualcosa di diverso da F o C. Se così hai fatto, la riga 40 termina il programma. Se invece hai immesso una C, il computer si porta automaticamente alla riga 50 per eseguire la conversione da Celsius a Fahrenheit.

A prima vista può sembrare esserci troppe istruzioni IF istruzioni per controllare quello che inserisci, ma questa è una buona pratica di programmazione per risparmiarti un bel po' di delusioni. Dovresti sempre cercare il modo che il tuo programma prossa prevedere tutte le possibilità.
Tornando all'esempio: non appena il programma riconosce quale tipo di conversione fare, vengono eseguiti i calcoli. Quindi il programma stampa la temperatura che hai inserito e l'equivalente della sua conversione. Il calcolo eseguito dal programma è solo pura matematica dato che usa la formula standard per la conversione della temperatura. Una volta finito e stampato il calcolo, il programma ritorna in ciclo sul quale ricominciare. Ecco una esecuzione d'esempio di questo programma:

```
CONVERTO DA FAHRENHEIT O CELSIUS (F/C): ? F
INSERISCI I GRADI FAHRENHEIT: 32
32 GRADI FAHRENHEIT = 0 GRADI CELSIUS

CONVERTO DA FAHRENHEIT O CELSIUS (F/C): ?
```

Una volta eseguito questo programma, potresti salvarlo su disco. Questo programma, come pure gli altri di questo manuale, può formare parte della tua libreria di programmi.

5.3. USO DELL'ISTRUZIONE GET PER IMMISSIONE DATI

GET ti permette di inserire un carattere alla volta dalla tastiera senza dover premere il tasto <RETURN>. In molti casi questo comando accelera veramente l'inserimento di dati.
Quando esegui un programma che ha una istruzione GET, alla sua variabile viene assegnato qualunque tasto premuto. Ecco un esempio:

```
 1 PRINT "(SHIFT-CLR/HOME)"
10 GET A$: IF A$ = "" THEN 10         Nessun spazio tra le virgolette
20 PRINT A$;
30 GOTO 10
```

La riga 1 pulisce lo schermo.
La riga 10 rimane in attesa di un qualsiasi tasto della tastiera. In pratica, il ciclo della riga 10 dice al computer di aspettare la pressione di un tasto qualsiasi prima di passare alla riga 20.
La riga 20 visualizza sullo schermo i tasti che premi.
La riga 30 rimanda il programma all'istruzione GET per acquisire un altro carattere. È importante ricordare che il carattere che immetti non verrà visualizzato a meno che tu non lo stampi sullo schermo come abbiamo fatto noi sulla riga 20.
L'istruzione IF nella riga 10 è molto importante. Poiché GET lavora di continuo, anche quando non premi un tasto (diversamente da INPUT che rimane in attesa della tua risposta), la seconda parte della riga 10 controlla di continuo la tastiera fino a quando non viene premuto un tasto.

Prova a togliere la seconda parte della riga 10 e guarda cosa succede.
Per fermare il programma premi i tasti <RUN/STOP> e <RESTORE>. Puoi facilmente riscrivere l'inizio del programma per la conversione della temperatura utilizzando GET al posto di INPUT. Se hai salvato questo programma, caricalo e modifica le righe 10 e 20 in questo modo:

```
10 PRINT "CONVERTO DA FAHRENHEIT O CELSIUS (F/C)"
20 GET A$: IF A$ = "" THEN 20
```

Questa modifica rende il programma più gradevole perché non succede niente fino a quando non immetti una delle due risposte (F o C) che selezionano il tipo di conversione. Se vuoi mantenere il programma, salvalo un'altra volta.

5.4. USO DI GET PER PROGRAMMARE I TASTI FUNZIONE

Come ricorderai, in un capitolo precedentei abbiamo detto che i tasti sul lato destro della tastiera (da F1 a F8) sono tasti funzione che puoi programmare per una varietà di compiti. Ecco come programmare un tasto funzione:

1) Usa una istruzione GET per leggere la tastiera.
2) Usa le istruzioni IF per confrontare il tasto che premi con il codice CHR$ del tasto funzione che vuoi utilizzare. Ciascun carattere della tastiera ha un numero CHR$ unico. Per esempio, il codice CHR$ di F1 è 133. L'Appendice F elenca i codici CHR$ di tutti i tasti.
3) Usa le istruzioni THEN per dire al computer cosa vuoi far fare al tasto funzione.

Quando esegui il programma, tutto quello che devi fare è premere un tasto funzione che hai programmato, ed il tasto seguirà le istruzioni che gli hai dato nell'istruzione THEN. Per esempio:

```
10 GET A$: IF A$ = "" THEN 10
20 IF A$ = CHR$(137) THEN PRINT CHR$(14)
30 IF A$ = CHR$(134) THEN PRINT "TANTI SALUTI"
```

La riga 10 dice al programma di assegnare il tasto premuto alla variabile A$. Come ricorderai dall'esempio precedente, il ciclo nella riga 10 controlla di continuo la tastiera in attesa di una immissione.
La riga 20 programma il tasto funzione 2, CHR$(137). La riga 20 dice al computer che, se premi il tasto funzione 2, di rendere A$ uguale a CHR$ (14).
CHR$ (14) è il commutatore sulla tastiera per passare dal maiuscolo al minuscolo. Quando esegui (RUN) questo programma, se premi F2 vedrai che i caratteri sullo schermo passeranno subito in questa modalità.
La riga 30 programma il tasto funzione 3, CHR$(134). Se premi F3 durante l'esecuzione del programma, la riga 30 dice al computer di assegnare ad A$ la stringa di caratteri TANTI SALUTI ed il carattere di controllo CHR$(13) che altro non è che il codice per il tasto <RETURN>.
I codici CHR$ dei tasti funzione sono:

F1 = CHR$(133)	F2 = CHR$(137)
F3 = CHR$(134)	F4 = CHR$(138)
F5 = CHR$(135)	F6 = CHR$(139)
F7 = CHR$(136)	F8 = CHR$(140)

La COMMODORE 64 PROGRAMMER'S REFERENCE GUIDE contiene maggiori informazioni riguardanti la programmazione dei tasti funzione. Puoi acquistare questa guida estesa dal tuo commerciante locale Commodore.

5.5. NUMERI CASUALI ED ALTRE FUNZIONI

Il 64 possiede anche delle funzioni interne che puoi utilizzare per eseguire particolari compiti. Le funzioni sono come programmi incorporati nel BASIC. Il vantaggio di queste funzioni interne è che non hai bisogno di immettere una quantità di istruzioni ogni volta che vuoi eseguire un calcolo specializzato. Anzi, tutto quello che devi fare è digitare il comando della funzione che vuoi, ed il computer farà tutto il resto.

Queste funzioni interne comprendono la rappresentazione di radici quadrate (SQR), rilevamento dei contenuti di una posizione di memoria (PEEK), generazione di numeri casuali (RND), ecc. L'Appendice C elenca tutte le funzioni disponibili sul tuo computer.

Una funzione con la quale puoi avere un po' di divertimento è la funzione per i numeri casuali, RND. Se progetti un gioco o un programma educativo, spesso dovrai programmare il tuo computer per formare numeri casuali. Per esempio, avrai bisogno di farlo per simulare il lancio di dadi. Naturalmente potresti scrivere un programma che possa generare questi numeri casuali, ma è molto più facile poterlo fare solo richiamando la funzione prescritta per i numeri casuali.

Per vedere come lavora RND, prova questo programmino:

```
NEW
10 FOR X = 1 TO 10
20 PRINT RND(1),             SE NON METTI QUESTA VIRGOLA, L'ELENCO DEI
30 NEXT                      NUMERI RIMANE IN UNA COLONNA (LA PRIMA)
```

Quando esegui questo programma sullo schermo si visualizzano:

```
.789280697          .664673958
.256373663          .0123442287
.682952381          3.90587279E-04
.402343724          .879300926
.158209063          .245596701
```

I tuoi numeri non sono uguali? Sarebbe incredibile se lo fossero perché il programma genera un elenco completamente casuale di dieci numeri. Se esegui il programma qualche altra volta, potrai vedere che i risultati saranno sempre diversi. Sebbene i numeri non abbiano un modello, sull'elenco visualizzato dal programma noterai qualche costante. Per prima cosa i risultati sono sempre tra 1 e 0, ma mai uguale ad 1 o 0. Seconda cosa, i numeri sono numeri reali (con cifre decimali). Se a questo punto cominciamo a simulare i lanci del dado, i risultati dati da questo programma non sono esattamente quello che vogliamo. A questo programma aggiungeremo altre funzioni per arrivare a ciò che vogliamo. Prima di tutto aggiungiamo al programma questa riga, che sostituisce la riga 20, ed eseguiamo il programma un'altra volta:

```
20 PRINT 6*RND(1),
RUN

3.60563664          4.52687513
5.48602315          1.09650123
3.10045018          4.39052168
3.91302584          5.06321506
2.32056144          4.10781302
```

Adesso abbiamo risultati maggiori di 1, ma ci sono ancora numeri reali. Per risolvere questo problema utilizzeremo un'altra funzione. La funzione INT converte i numeri reali in numeri interi. Per cui sostituiamo ancora la riga 20:

```
20 PRINT INT(6*RND(1),
RUN

2         3         1         0
2         4         5         5
0         1
```

Adesso siamo proprio vicini al nostro scopo, ma noterai che la serie dei numeri va da 0 al 5 e non dall'1 al 6. Come ultimo passo andiamo a sostituire la riga 20 un'altra volta:

```
20 PRINT INT(6*RND(1))+1
```

Ed è adesso, quando eseguirai il programma, che otterrai i risultati voluti. Quando vuoi generare un campo di numeri reali invece di numeri interi, la formula è leggermente diversa perché devi sottrarre il limite più basso del campo al limite più alto. Per esempio, puoi generare numeri casuali tra 1 e 25 immettendo:

```
20 PRINT RND(1)*(25-1)+1
```

La formula generica per generare numeri casuali in un certo campo è:

NUMERO = RND(1) * (LIMITE SUPERIORE - LIMITE INFERIORE) + LIMITE INFERIORE

5.6. GIOCO DELL'INDOVINO

Ecco un gioco che utilizza i numeri casuali. Questo gioco non solo usa la funzione RND, ma introduce anche qualche altra teoria di programmazione. Quando esegui questo programma, il computer genererà un numero casuale, NM, che dovrai cercare di indovinare in meno mosse possibili.

```
NEW
  1 REM GIOCO PER INDIVINARE IL NUMERO
  2 PRINT "(CLR/HOME
  5 INPUT "IMMETTI IL LIMITE SUPERIORE DA INDOVINARE "; LI
 10 NM = INT(LI*RND(1))+1
 15 CN = 0
 20 PRINT "HO IL NUMERO.": PRINT
 30 INPUT "PROVA AD INDOVINARLO "; GU
 35 CN = CN + 1
 40 IF GU > NM THEN PRINT "IL MIO NUMERO E' INFERIORE." : PRINT : GOTO 30
 50 IF GU < NM THEN PRINT "IL MIO NUMERO E' SUPERIORE.": PRINT : GOTO 30
 60 IF GU = NM THEN PRINT "GRANDE! E' IL MIO NUMERO"
 65 PRINT "IN SOLE"; CN ;"PROVE." : PRINT
 70 PRINT "VUOI PROVARE UN'ALTRA VOLTA (S/N)?"
 80 GET AN$ : IF AN$ = "" THEN 80
 90 IF AN$ = "S" THEN 2
100 IF AN$ <> "N" THEN 70
110 END
```

Indica senza spazio dopo le virgolette

All'inizio del programma puoi indicare la grandezza del numero. In seguito, starà a te indovinare quale sia il numero. Dopo l'esecuzione segue una spiegazione.

```
IMMETTI IL LIMITE SUPERIORE DA INDOVINARE? 25
HO IL NUMERO.

PROVA AD INDOVINARLO ? 15
IL MIO NUMERO E' SUPERIORE.

PROVA AD INDOVINARLO ? 20
IL MIO NUMERO E' INFERIORE.

PROVA AD INDOVINARLO ? 19
GRANDE! E' IL MIO NUMERO
IN SOLE 3 PROVE.
```

```
VUOI PROVARE UN'ALTRA VOLTA (S/N)?
```

L'istruzione IF/THEN (righe 40-60) confronta il tuo tentativo col numero casuale (NM) generato dalla riga 10. Se il tuo tentativo è sbagliato, il programma ti avverte di stare più alto o più basso del valore di NM. Ogni volta che fai un tentativo, la riga 35 aggiunge 1 a CN il quale, in pratica, è un contatore che tiene traccia di quanti tentativi avrai fatto prima di indovinare il numero. Lo scopo di questo gioco, naturalmente, è quello di indovinare il numero in meno tentativi possibili. Quando inserisci la risposta giusta, il programma visualizza il messaggio, `GRANDE! E' IL MIO NUMERO`, e ti dice quanti tentativi hai fatto.
Ricorda che, ogni volta che esegui il programma, il numero casuale sarà sempre diverso.
Al programma potresti aggiungere alcune righe per indicare anche il limite inferiore dei numeri generati da questo gioco.

SUGGERIMENTI ALLA PROGRAMMAZIONE:
Nelle righe 40 e 50, i due punti separano istruzioni multiple su una singola riga. Questo sistema non solo fa risparmiare tempo di battitura, ma riduce anche spazio in memoria.
Nota anche che le istruzioni IF/THEN, in queste due righe, stampano qualcosa prima di diramare su un'altra riga.

5.7. Il TUO ROTOLAMENTO

Il programma seguente simula il lancio di due dadi. Con questo piccolo programma puoi giocarci da solo oppure utilizzarlo come parte di un gioco più ampio.

```
 5 PRINT "CERCHI FORTUNA?"
10 PRINT "DADO ROSSO = "; INT(RND(1)*6)+1
20 PRINT "DADO BIANCO = "; INT(RND(1)*6)+1
30 PRINT "PREMI SPAZIO PER UN ALTRO TIRO": PRINT
40 GET A$: IF A$ = "" THEN 40
50 IF A$ = CHR$(32) THEN 10
60 END
```

Da quello che hai imparato sul BASIC ed i numeri casuali, cerca di capire quello che sta facendo questo programma. Come puoi ricordare dal paragrafo sulla programmazione dei tasti funzione, CHR$(32) è il codice di carattere per la barra dello spazio.

5.8. GRAFICA CASUALE

Come ultima nota sui numeri casuali, e come introduzione al disegno di grafica, cerca di immettere ed eseguire questo programma:

```
10 PRINT "(SHIFT-CLR/HOME)"
20 PRINT CHR$(205.5 + RND(1));
30 GOTO 20
```

La funzione CHR$ (stringa di carattere) ti dà un carattere in base ad un numero di codice standard da 0 a 255. Ogni carattere che può stampare il 64 è codificato in questo modo. L'Appendice F elenca i codici CHR$ di tutti i tasti.
Un sistema veloce di scoprire il codice di ogni carattere è di utilizzare la funzione ASC (per il codice ASCII standard). Batti:

```
PRINT ASC("X")
```

X è il carattere che stai esaminando. X può essere un qualsiasi carattere stampabile, compresi i caratteri grafici. Il carattere deve essere racchiuso fra virgolette. Ecco un esempio:

```
PRINT ASC("G")
71
```

La funzione CHR$ è esattamente l'opposto della funzione ASC.

```
PRINT CHR$(71)
G
```

Se inserisci:

```
PRINT CHR$(205);CHR$(206)
```

il computer visualizza i due grafici sul lato destro dei tasti M e N, che sono i caratteri utilizzati nel piccolo programma che crea il labirinto e che hai appena provato. La formula 205.5+ RND (1) dice al computer di prendere un numero casuale tra 205,5 e 206,5. C'è il cinquanta per cento di possibilità che il numero casuale sia sopra o sotto il 206. Poiché la funzione CHR$ ignora i valori frazionati, i caratteri visualizzati sullo schermo saranno suddivisi fra il codice 205 ed il codice 206.
Con questo programma puoi provare ad aggiungere o sottrarre un paio di decimi dal 205.5 dandoti maggiori probabilità di visualizzare uno solo dei due caratteri.

6. COLORE E GRAFICA

6.1. COME UTILIZZARE IL COLORE E LA GRAFICA SUL TUO COMPUTER

Fin qui questo libro ha presentato alcune delle sofisticate capacità di calcolo del tuo 64. Tuttavia, una delle caratteristiche più eccitanti del tuo nuovo computer è la sua notevole capacità di presentare 16 colori diversi ed un bel po' di grafiche differenti.
Hai già potuto vedere una dimostrazione molto semplice della grafica nel programma della palla che rimbalza e nel programma del labirinto in fondo al capitolo precedente. Questo capitolo ti introduce verso nuovi concetti che spiegano la programmazione di grafica e colore, e che ti suggeriscono le idee per la creazione dei tuoi propri giochi e delle animazioni avanzate.

6.2. STAMPA DEI COLORI

Quando nel Capitolo 1 hai imparato la posizione dei colori, hai scoperto che puoi cambiare colore al testo semplicemente tenendo premuto il tasto <CTRL> e premendo uno del tasti per il colore.
Il 64 offre tutta una serie di 16 colori, e sebbene sui tasti del colore ne siano presenti solo otto, puoi avere un'altra serie di otto colori tenendo premuto il tasto <C=> e premendo uno dei tasto per il colore. Qui c'è un elenco dei colori:

TASTIERA	COLORE	A SCHERMO	TASTIERA	COLORE	A SCHERMO
<CTRL> <1>	NERO	Caretteri grafici omessi	<C=> <1>	ARANCIO	Caretteri grafici omessi
<CTRL> <2>	BIANCO		<C=> <2>	MARRONE	
<CTRL> <3>	ROSSO		<C=> <3>	ROSSO CHIARO	
<CTRL> <4>	CIANO		<C=> <4>	GRIGIO 1	
<CTRL> <5>	PORPORA		<C=> <5>	GRIGIO 2	
<CTRL> <6>	VERDE		<C=> <6>	VERDE CHIARO	
<CTRL> <7>	BLU		<C=> <7>	BLU CHIARO	

<CTRL> <8>	GIALLO	<C=> <8>	GRIGIO 3

Quando ti abbiamo mostrato il programma della palla che rimbalza, nell'ultimo capitolo, hai visto che i comandi da tastiera, come i movimenti per il cursore, possono essere scritti nelle istruzioni PRINT. Allo stesso modo puoi anche cambiare colore al testo dei tuoi programmi.

Digita NEW e fai delle prove cambiando i colori. Tieni premuto il tasto <CTRL> e contemporaneamente premi il tasto <1>; rilascia tutti e due i tasti e premi il tasto <R>. Adesso ripremi il tasto <CTRL> insieme al tasto <2> tasto; rilasciali e premi il tasto <A>. Prosegui così alternando <CTRL> ed il numero del colore, con le lettere per formare la parola RAINBOW; in questo modo:

```
10 PRINT " R A I N B O W"
           ^ ^ ^ ^ ^ ^ ^
    <CTRL><1 2 3 4 5 6 8>
```

Ricorderai che nell'istruzione PRINT i controlli del cursore si presentano come caratteri grafici e che vengono raffigurati come caratteri grafici. La tabella dei colori, mostrata sopra, fa vedere i caratteri grafici che si presentano per ogni colore. A causa di questa rappresentazione dei caratteri grafici, una volta digitati la tua istruzione PRINT apparirà alquanto strana ma, quando eseguirai il programma, vedrai che verrà visualizzato solo il testo del messaggio. Le lettere nel messaggio si presenteranno dello stesso colore digitando i controlli di colore nell'istruzione PRINT. Adesso prova a formare altri esempi di tua fantasia mescolando i vari controlli di colore e mettendoli all'interno di un'unica istruzione PRINT. Non dimenticarti della seconda serie di colori che puoi avere premendo il tasto <C=> insieme ad un tasto del colore.

> SUGGERIMENTO:
> Dopo aver eseguito un programma in cui hai fatto modifiche sulla modalità (inversa) o sul colore, noterai che sia il prompt READY che qualsiasi ulteriore testo digitato in seguito verranno visualizzati con l'ultimo colore o modalità che hai fatto sulla riga del programma. Per tornare alla visualizzazione normale, premi questi tasti insieme:
> RUN/STOP e RESTORE

6.3. CODICI CHR$ PER I COLORI

Prima di cominciare a leggere questo paragrafo, dai un'occhiata all'Appendice F che elenca i codici CHR$ di tutti i tasti della tastiera. Esaminando l'elenco dei codici CHR$, probabilmente avrai notato che ogni colore ha un codice unico, proprio come tutti gli altri tasti ed i controlli della tastiera. Se stampi questi codici utilizzando la funzione CHR$ citata nell'ultimo capitolo, otterrai gli stessi risultati immettendo <CTRL> o <C=> ed il tasto colore in una istruzione PRINT.
Per fare un esempio, prova questo:

```
NEW
10 PRINT CHR$(147): REM <CLR/HOME>
20 PRINT CHR$(28);"CHR$(28) MI CAMBIA IN?"

RUN
CHR$(28) MI CAMBIA IN?
```

Quando esegui (RUN) questo programma, verrà prima vuotato lo schermo e poi visualizzato il messaggio della riga 20. Adesso il testo dovrà essere rosso. In molti casi troverai che è molto più facile utilizzare la funzione CHR$ per cambiare i colori, specialmente quando fai le tue prove. La pagina successiva mostra un altro modo per avere un arcobaleno di colori. Dato che nel programma ci sono molte righe simili (dalla 40 alla 110), usa i tasti di editazione per evitare un bel po' di battitura. Per rinfrescarti la memoria sulle procedure di editazione, consulta le note alla fine del listato del programma.

```
NEW

  1 REM BARRE AUTOMATICHE DI COLORE
  5 PRINT CHR$(147): REM CHR$(147) = CLR/HOME
 10 PRINT CHR$(18); "     ";: REM BARRA INVERSA
 20 CL = INT(8*RND(1))+1
 30 ON CL GOTO 40,50,60,70,80,90,100,110
 40 PRINT CHR$(5);: GOTO 10
 50 PRINT CHR$(28);: GOTO 10
 60 PRINT CHR$(30);: GOTO 10
 70 PRINT CHR$(31);: GOTO 10
 80 PRINT CHR$(144);: GOTO 10
 90 PRINT CHR$(156);: GOTO 10
100 PRINT CHR$(158);: GOTO 10
110 PRINT CHR$(159);: GOTO 10
```

Digita come al solito le righe dalla 1 alla 40. La visualizzazione dovrà essere questa:

```
 1 REM BARRE AUTOMATICHE DI COLORE
 5 PRINT CHR$(147): REM CHR$(147) = CLR/HOME
10 PRINT CHR$(18); "     ";: REM BARRA INVERSA
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
40 PRINT CHR$(5);: GOTO 10
_
```

NOTE PER L’EDITAZIONE:

Usa il tasto <CRSR-UP> per mettere in posizione il cursore sulla riga 40, quindi digita 5 sul 4 del 40. Adesso usa il tasto <CRSR-RIGHT> per arrivare sul 5 fra parentesi del CHR$. Premi <SHIFT> e <INST/DEL> per allargare di uno spazio ed inserire 28. Adesso basta premere <RETURN> col cursore sulla riga. La visualizzazione dovrà presentarsi così:

```
NEW

 1 REM AUTOMATIC COLOR BARS
 5 PRINT CHR$(147): REM CHR$(147) = CLR/HOME
10 PRINT CHR$(18); "     ";: REM REVERSE BAR
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
50 PRINT CHR$(28);: GOTO 10
```

Non preoccuparti della riga 40; è ancora là, come puoi vedere LISTando il programma. Segui gli stessi passi per modificare la riga 40 con un altro numero di riga ed il codice CHR$ fino a quando non hai inserito tutte le righe restanti. Come ultimo controllo, LISTa tutto il programma per assicurarti di avere tutte le righe necessarie prima di eseguirle.

Probabilmente avrai capito il programma delle barre colorate tranne la riga 30. Ecco una breve spiegazione di come lavora questo programma.

La riga 5 stampa il codice CHR$ del <CLR/HOME>.

La riga 10 attiva la modalità inversa e stampa 5 spazi che, in questa modalità, diventano pieni anzichè vuoti. La prima volta che il programma entra in questa riga, la barra sarà blu chiara, il colore della normale visualizzazione della schermata.

La riga 20 si serve della funzione RaNDom per selezionare casualmente un colore tra 1 e 8.

La riga 30 usa una variazione dell'istruzione IF/THEN, chiamata ON/GOTO, che permette al programma di scegliere da un elenco di numeri di riga, su uno dei quali dovrà saltare il programma in base al valore di una variabile. Se la variabile di ON (in questo caso CL) ha il valore 1, il programma salterà al primo numero di riga elencato (qui, la riga 40). Se la variabile ha il valore 2, il programma salterà al secondo numero di riga elencato, e così via..

Le righe dalla 40 alla 110 convertono i colori a chiave casuale nel giusto codice CHR$ per quel colore e fanno ritornare il programma alla riga 10 per stampare i 5 spazi pieni con quel colore. Dopo di ché il processo ricomincia un'altra volta.
Guarda se capisci come presentare le barre in 16 colori casuali. Amplia l'ON/GOTO per gestire gli ulteriori colori ed aggiungi i codici CHR$ che rimangono.

6.4. COME UTILIZZARE LE PEEK E LE POKE

Le PEEK e le POKE ti permettono di esaminare la memoria del tuo computer e di metterci cose esattamente dove le vuoi.
Ricorderai che nel Capitolo 4 abbiamo spiegato le variabili come fossero piccole fessure nella memoria del computer, con il nome della variabile come indirizzo della fessura. Bene, immagina alcune fessure del computer, più particolarmente definite, che indichino delle posizioni ben precide della memoria e che, come numero, abbiano i loro indirizzi.
Il tuo 64 esamina queste posizioni di memoria per vedere quali dovranno essere i colori del fondo e del bordo dello schermo, quali caratteri visualizzare sullo schermo e dove visualizzarli, ecc.
Da parte tua puoi modificare i colori dello schermo, definire e muovere oggetti e perfino creare musica inserendo un valore diverso nelle ben precise fessure della memoria. Immagina le fessure della memoria come a qualcosa di simile:

```
+-----------+     +------------+      +------------+      +------------+
|   53280   |     |   53281    |      |   53282    |      |   53283    |
|     2     |     |     1      |      |            |      |            |
+-----------+     +------------+      +------------+      +------------+
   COLORE            COLORE
   BORDO             FONDO
```

Le prime due fessure sono le posizioni della memoria per i colori del bordo e del fondo sul tuo schermo. Nella casella che riguarda il colore del bordo noi abbiamo messo un 2, il valore per il ROSSO, mentre nella casella per il colore del fondo abbiamo messo un 1, il valore per il BIANCO. Adesso prova a digitare questo:

```
POKE 53281,7 <RETURN>
```

Il colore del fondo del tuo schermo cambierà in giallo perché abbiamo messo il valore 7, appunto del giallo, nella posizione che controlla il colore del fondo.
Prova ad inserire altri valori nella posizione del colore del fondo e guarda quali risultati ricevi.Qui c'è un elenco dei valori di POKE per ciascun colore disponibile sul tuo 64:

0	NERO	8	ARANCIO
1	BIANCO	9	MARRONE
2	ROSSO	10	ROSSO CHIARO
3	CIANO	11	GRIGIO 1
4	PORPORA	12	GRIGIO 2
5	VERDE	13	VERDE CHIARO
6	BLU	14	BLU CHIARO
7	GIALLO	15	GRIGIO 3

Ecco un breve programma che puoi utilizzare per visualizzare varie combinazioni di colore per il bordo e lo sfondo:

```
NEW

10 FOR BA = 0 TO 15
20 FOR BO = 0 TO 15
30 POKE 53280, BO
40 POKE 53281, BA
50 FOR X = 1 TO 2000: NEXT X
60 NEXT BO : NEXT BA
```

Questo programma usa due semplici cicli per inserire (POKE) valori diversi e modificare i colori per lo fondo ed il bordo. La riga 50 contiene un ciclo di RITARDO che, appunto, rallenta un po' il programma per permetterti di vedere questi cambi di colore.
Se sei curioso di controllare quale sia il valore attualmente nella posizione di memoria per il colore del fondo, prova questo:

```
? PEEK (53280) AND 15
```

PEEK esamina un byte intero, ma i colori utilizzano solo la metà di un byte, chiamato nybble. Per dare una sbirciatina (PEEK) proprio a questo nybble, alla tua istruzione PEEK devi aggiungere AND 15. Se hai usato questa PEEK dopo aver eseguito il programma precedente, come risposta avrai ricevuto 15 perché l'ultimo colore inserito per il bordo è stato il GRIGIO 3, che appunto è il 15.
Generalmente PEEK ti permette di osservare quale valore si trovi attualmente in una ben precisa fessura della memoria. Prova ad aggiungere questa riga al tuo programma per visualizzare i valori del BORDO e del FONDO una volta eseguito il programma.

```
25 PRINT CHR$(147); "BORDO = "; PEEK(53280) AND 15,
     "FONDO = "; PEEK(53281) AND 15
```

6.5. GRAFICA DI SCHERMO

Fino ad ora, quando hai stampato le informazioni, il computer le ha gestito in modo sequenziale: un carattere stampato dietro l'altro, a partire dalla posizione attuale del cursore, tranne quando nella formattazione del PRINT gli hai ordinato di andare a capo o utilizzato una virgola. Tu puoi stampare dati in una posizione ben precisa, a partire da una posizione nota sullo schermo, e stampare l'esatta quantità di controlli del cursore per formattare la visualizzazione. Questo sistema, però, porta via tempo e passi di programma.
Ma proprio come ci sono certe posizioni nella memoria del 64 per controllare il colore, ci sono anche posizioni che puoi utilizzare per controllare le posizioni dello schermo.

6.6. MAPPA DELLA MEMORIA DI SCHERMO

La schermata del 64 può contenere 1000 caratteri (40 colonne per 25 righe), per cui sono impostate 1000 locazioni di memoria per rappresentare quello che si trova sullo schermo. Immagina lo schermo come una griglia, 40 per 25, con ogni cella che indica una locazione di memoria. Ciascuna locazione di memoria può contenere uno dei 256 diversi caratteri visualizzati dal 64 (consulta l'Appendice E). Ciascuno di questi 256 caratteri viene rappresentato da un numero da 0 a 255. Se inserisci (POKE) il valore di un carattere dentro una ben precisa posizione della memoria di schermo, quel carattere verrà visualizzato in quella particolare posizione dello schermo. Qui c'è una griglia che rappresenta il tuo schermo, completa dei numeri di ciascuna locazione di memoria.

```
                           COLONNA                       1063
      0          10          20          30          39 /
     +------------------------------------------------------/
1024 |                                                      |  0
1064 |                                                      |
1104 |                                                      |
1144 |                                                      |
1184 |                                                      |
1224 |                                                      |
1264 |                                                      |
1304 |                                                      |
1344 |                                                      |
1384 |                                                      |
1424 |                                                      | 10
1464 |                                                      |
```

```
1504 |                                                       |   RIGA
1544 |                                                       |
1584 |                                                       |
1624 |                                                       |
1664 |                                                       |
1704 |                                                       |
1744 |                                                       |
1784 |                                                       |
1824 |                                                       | 20
1864 |                                                       |
1904 |                                                       |
1944 |                                                       |
1984 |                                                       | 24
     +-------------------------------------------------------\
                                                              \
                                                               2023
```

Normalmente la memoria di schermo del 64 inizia dalla locazione 1024 e finisce alla locazione 2023. La locazione 1024 è l'angolo superiore sinistro dello schermo. La locazione 1025 è la posizione del successivo carattere verso destra, e così via. La locazione 1063 è la posizione più a destra della prima riga. Seguendo l'ultimo carattere di una fila, la locazione successiva è il carattere più a sinistra della fila successiva, quella sotto.

Supponi di voler controllare una palla rimbalzante sullo schermo. La palla si trova al centro dello schermo, colonna 29, fila 12. La formula per calcolare la locazione di memoria sullo schermo è:

```
PUNTO = 1024 + X + 40 * Y <---- fila
               ^--------- colonna
```

dove X è la colonna ed Y la fila. Quindi, la locazione di memoria della palla sarà:

```
PUNTO = 1024 + 20 + 480 <-- fila (40x12)
PUNTO = 1524   ^------- colonna
```

Pulisci lo schermo con <SHIFT> e <CLR/HOME> e digita:

```
POKE 1524,81 <--- codice del carattere
       ^------ locazione
```

Questa istruzione POKE crea un palla che appare al centro dello schermo. Hai inserito un carattere direttamente nella memoria di schermo senza usare l'istruzione PRINT. Tuttavia, ancora non puoi vedere la palla perché è dello stesso colore del fondo dello schermo.

6.7. MAPPA DELLA MEMORIA PER IL COLORE

Puoi cambiare il colore della palla che è apparsa, modificando un'altra serie di memorie. Digita:

```
POKE 55796,2 <--- colore
       ^---- locazione
```

Questo comando cambierà in rosso il colore della palla.

Ogni punto sullo schermo del 64 ha DUE locazioni di memoria: uno per il codice del carattere ed uno per il codice del colore. La mappa della memoria del colore inizia dalla locazione 55296 (l'angolo superiore sinistro) e continua per 1000 locazioni. Per i codici dei colori, basta utilizzare gli stessi, da 0 a 15, che servivano a modificare i colori del bordo e del fondo o del carattere.

Per avere le locazioni in cui inserire (POKE) i colori, possiamo modificare la formula usata per calcolare le locazioni della memoria per lo schermo. Qui c'è la nuova formula:

```
COLORE STAMPATO = 55296 + X + 40 * Y <--- fila
                          ^--------- colonna
```

6.8. ANCORA PALLE CHE RIMBALZANO

Ecco il programma corretto, della palla che rimalza, che stampa direttamente sullo schermo utilizzando le POKE al posto dei controlli di cursore nell'istruzioni PRINT. Quando esegui questa versione, vedrai che è molto più flessibile del programma precedente e servirà da introduzione alla programmazione di animazioni più elaborate.

```
NEW

 10 POKE 53281,1: PRINT"<CTRL/WHITE><SHIFT CLR/HOME>"
 20 POKE 53280,7: POKE 53281,6
 30 X = 1 : Y = 1
 40 DX = 1 : DY = 1
 50 POKE 1024 + X + 40 * Y, 81
 60 FOR T = 1 TO 10 : NEXT
 70 POKE 1024 + X + 40 * Y, 32
 80 X = X + DX
 90 IF X <= 0 OR X >= 39 THEN DX = -DX
100 Y = Y + DY
110 IF Y <= 0 OR Y >= 24 THEN DY = -DY
120 GOTO 50
```

La riga 10 imposta su bianco il colore del cursore e pulisce lo schermo. NOTA: Liberando lo schermo sui 64-NTSC si imposta su bianco la RAM del colore ma sui 64-PAL la RAM del colore viene impostata sull'attuale colore del fondo (qui bianco).
La riga 20 imposta su blu il colore del fondo e su giallo il colore del bordo.
Le variabili X ed Y, nella riga 30, tengono traccia della posione attuale della palla (fila e colonna). Le variabili DX e DY, nella riga 40, rappresentano la direzione di movimento orizzontale e verticale della palla.
Quando viene aggiunto un +1 al valore di X, la palla si muove verso destra; quando viene aggiunto un -1, la palla si muove verso sinistra. Un +1 aggiunto a Y fa muovere la palla in basso di una riga, mentre un -1 aggiunto a Y fa muovere la palla in alto di una riga.
La riga 50 inserisce la palla sullo schermo alla posizione attuale di X e Y.
La riga 60 è un ciclo di ritardo che viene aggiunto per mantenere la palla sullo schermo quanto basta per permettere di vederla. Senza questo ciclo la palla si muoverebbe troppo veloce per poterla inquadrare.
La riga 70 cancella la palla inserendo uno spazio (codice 32) nel punto in cui si trovava sullo schermo.
La riga 80 aggiunge ad X il coefficiente di direzione.
La riga 90 fa un controllo per vedere se la palla ha raggiunto una delle pareti laterali, e se qui c'è un rimalzo ne inverte la direzione. Le righe 100 e 110 fanno la stessa cosa per le pareti superiore ed inferiore.
La riga 120 rispedisce la palla alla visualizzazione e la sposta un'altra volta.
Cambiando il codice sulla riga 50 puoi sostituire la palla con un qualsiasi altro carattere: da 81 ad un altro codice del carattere.
Se metti a 0 DX o DY, la palla rimbalza diritta invece che diagonalmente.
A questo programma possiamo anche aggiungere un po' di ingegno. Fino ad ora l'unica cosa che hai controllato è se la palla usciva dai limiti dello schermo. Prova ad aggiungere al programma le seguenti righe:

```
 21 FOR L = 1 TO 10
 25 POKE 1024 + INT(RND(1)*1000), 160  <---(SPAZIO INVERSO)
 27 NEXT L
115 IF PEEK(1024 + X + 40*Y) = 166 THEN DX = -DX : GOTO 80
```

Le righe dalla 21 alla 27 mettono sullo schermo 10 blocchi in posizioni casuali. La riga 115 controlla (PEEK) per vedere se la palla si trova a rimbalzare in un blocco e, se vero, modifica la direzione della palla.

7. INTRODUZIONE ALLE FIGURE GRAFICHE (SPRITE)

Nei capitoli precedenti ti abbiamo mostrato come utilizzare i simboli grafici nelle istruzioni PRINT per creare animazioni ed altri effetti visivi. Nel capitolo 6 ti abbiamo mostrato anche come inserire (POKE) i codici di carattere in particolari locazioni della memoria di schermo che inseriscono i caratteri direttamente sullo schermo, nel punto esatto che hai scelto. In entrambi i casi devi creare gli oggetti con i simboli grafici disponibili e perdendo un bel po' di tempo per programmarli. In effetti, quando vuoi muovere gli oggetto, devi utilizzare una quantità di istruzioni di programma per tenere traccia dell'oggetto e muoverlo da un posto all'altro. E qualche volta la forma e la risoluzione degli oggetti non sono praticamente come li vorresti a causa delle limitazioni dei simboli grafici. Puoi eliminare un bel po' di questi problemi se, nelle sequenze animate, utilizzi le figure grafiche. Una figura grafica (o sprite) è un oggetto programmabile, ad alta risoluzione, che con i comandi del BASIC puoi creare in qualsiasi forma. Tutto quello che devi fare per muovere l'oggetto è semplicemente quello di far sapere al computer la posizione in cui vorrai far andare lo sprite. Il computer penserà a tutto il resto. Tuttavia questo non è tutto quello che puoi fare con gli sprite. Per esempio, puoi modificarne il colore, puoi sapere quando un oggetto collide con un altro, puoi farli passare davanti e dietro a ogni altro e puoi facilmente espanderne la dimensione.
Prima di poter utilizzare gli sprite, però, devi imparare qualche altro particolare che riguarda il tuo 64 ed il modo in cui gestisce i numeri. Non sono cose difficili, basta seguire gli esempi e sarai in grado di far fare ai tuoi sprite cose sorprendenti da subito.

7.1. BIT E BYTE

Prima che tu possa utilizzare gli sprite, è importante che tu capisca alcune cose generiche su come lavorano i computer. Nel sistema decimale noi contiamo in "decine" usando valori da 0 a 9. Quando una particolare posizione eccede il suo valore massimo di 9, esso ricomincia da zero e riporta un uno nella posizione successiva (alla sua sinistra). Per esempio: il numero 64 significa 6 x (10) + 4 x (1). La posizione di ciascuna cifra è importante. Il valore 64 viene calcolato come segue:

$$6 \times 10^1 + 4 \times 10^0$$

NOTA: Qualsiasi numero elevato a potenza zero è uguale a 1.

I computer memorizzano le informazioni come una serie di cariche elettriche che rappresentano gli 1 e gli 0. Ogni cella della memoria contiene un modello di otto uni e zeri chiamati cifre binarie o BIT. Queste celle, che comprendono tutti gli otto bit, sono chiamate BYTE. Un bit, che è la quantità più piccola dell'informazione che possa memorizzare un computer, può essere ACCESO dandogli un valore di 1, o SPENTO con un valore di 0. Quando inserisci le informazioni nel computer attraverso la tastiera, le pressioni sui tasti vengono convertite in modelli ad 8 bit di uni e zeri, e trasferiti in memoria. Le regole dell'aritmetica binaria sono molto più semplici degli altri sistemi poiché le cifre possono avere solo due valori, 0 o 1. Come spiegato nell'esempio precedente, il sistema decimale usa la base del 10, mentre il sistema binario usa la base del 2.
Un bit può contenere solo una delle due combinazioni, o 0 o 1. Con due bit abbiamo quattro possibili combinazioni di 1 e 0 (2^2) e con tre bit otto possibili combinazioni (2^3). La tabella che segue mostra il campo dei valori.

Quantità di BIT	Quantità di valori	Combinazioni possibili	
1	2^1 = 2	ON OFF	1 0
2	2^2 = 4	ON e ON ON e OFF OFF e ON OFF e OFF	1 1 1 0 0 1 0 0

3	2^3 = 8	ON e ON e ON	1 1 1
		ON e ON e OFF	1 1 0
		ON e OFF e ON	1 0 1
		ON e OFF e OFF	1 0 0
		OFF e ON e ON	0 1 1
		OFF e OFF e ON	0 0 1
		OFF e ON e OFF	0 1 0
		OFF e OFF e OFF	0 0 0

Come puoi vedere, la quantità delle combinazioni è data dal 2 (cioè la base) elevata alla potenza per la quantità di bit. Per un BYTE, o otto bit, puoi memorizzare 256 valori diversi, cioè 2^8.
Quando tutti gli otto bit sono SPENTI, cioè impostati su 0, il byte contiene un valore di zero. Quando tutti gli otto bit sono ACCESI, cioè impostati su 1, il byte ha un valore di 255. Da notare che il campo delle combinazioni è compreso da 0 a 255.
È possibile convertire in valore decimale qualsiasi numero binario semplicemente aggiungendo quelle potenze di due in cui è stato impostato un bit. L'esempio qui sotto spiega come venga trasformato in valore binario il valore decimale di 181:

POSIZIONI BINARIE	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
EQUIVALENTE DECIMALE	128	64	32	16	8	4	2	1
VALORI DEI BIT	1	0	1	1	0	1	0	1

Sommando i valori dei bit ACCESI si avrà:

2^7 + 2^5 + 2^4 + 2^2 + 2^0

oppure 128 + 32 + 16 + 4 + 1 = 181

La seguente è una tabella che mostra la conversione da binario a decimale. Uno zero indica che il bit è SPENTO, un 1 mostra che il bit è ACCESO. Per calcolare il valore di tutto il byte aggiungere il valore decimale ad ogni bit ACCESO.

CONVERSIONE DA BINARIO A DECIMALE

Valore del Decimale								
128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	1	2^0
0	0	0	0	0	0	1	0	2^1
0	0	0	0	0	1	0	0	2^2
0	0	0	0	1	0	0	0	2^3
0	0	0	1	0	0	0	0	2^4
0	0	1	0	0	0	0	0	2^5
0	1	0	0	0	0	0	0	2^6
1	0	0	0	0	0	0	0	2^7

SUGGERIMENTO:
La conversione di numeri binari con i loro valori decimali è la base per la creazione dei dati per rappresentare e gestire gli sprite. Qui sotto c'è un programma per fare la conversione.
Poiché con molta probabilità utilizzerai spesso questo programma, ti conviene digitarlo e salvarlo.

```
 5 REM CONVERSIONE DA BINARIO A DECIMALE

10 INPUT "INSERISCI UN NUMERO BINARIO AD 8 BIT :";A$

12 IF LEN(A$) <> 8 THEN PRINT "HO DETTO 8 BIT..." : GOTO 10
15 TL = 0 : C = 0
20 FOR X = 8 TO 1 STEP -1 : C = C + 1
30 TL = TL + VAL(MID$(A$,C,1))*2^(X-1)
40 NEXT X
50 PRINT A$; " BINARIO = "; TL ;" DECIMALE"
60 GOTO 10
```

Alla riga 10 ti viene chiesto di inserire un numero binario che entrerà nella variabile di stringa A$.
La riga 12 usa la funzione LEN (lunghezza) per assicurarsi che tu abbia immesso 8 cifre binarie. In caso negativo, te lo richiede ripetendo la riga 10.
Nella riga 15 TL tiene traccia del valore decimale del numero binario, e C indica quale bit è da processare via via che il programma attraversa il ciclo.
La riga 30 aggiorna il valore di TL. L'Appendice spiega le funzioni VAL e MID$.
La riga 50 stampa i valori binario e decimale del byte. La riga 60 fa tornare il programma all'inizio.

7.2. CREAZIONE DI UNO SPRITE

Registri degli Sprite

Prima di proseguire devi sapere come fa il COMMODORE 64 a gestire gli sprite. Gli sprite sono gestiti da un particolare circuito integrato, all'interno del tuo COMMODORE 64, chiamato VIC II. Questo microcircuito contiene un serie di speciali byte, chiamati REGISTRI, che sono specializzati proprio per la gestione degli sprite. Ciascun registro esegue una funzione a parte. Per esempio, l'ENABLE REGISTER controlla se uno sprite è attivo o inattivo, mentre gli EXPAND REGISTER controllano la dimensione di uno sprite. Quando lavori con gli sprite, pensa ad un registro come un byte con una funzione ben precisa. I registri di cui parleremo in questo capitolo sono elencati qui sotto:

```
REGISTRO No.  DESCRIZIONE
------------------------------------
   0-15       POSIZIONAMENTO DELLO SPRITE
     16       MOVIMENTO EXTRA
     21       ABILITAZIONE (ON/OFF)
     23       ESPANZIONE (VERTICALE)
     27       PRIORITA'
     28       SELEZIONE MULTI-COLORE
     29       ESPANSIONE (ORIZZONTALE)
  37-38       MULTI-COLORI
  39-46       COLORE
```

Ognuno di questi registri, o byte, è stato assegnato ad una locazione ben precisa nella memoria del computer, a cominciare dalla locazione 53248. Questo è l' 'indirizzo di base' del microcircuito VIC II. Per accedere ai singoli registri, è più facile assegnare ad una variabile il valore dell'indirizzo di partenza e poi aggiungerci il numero del registro; per es. V= 53248: POKE V+21,255. Questi due

comandi inseriranno il valore 255 nel registro 21. Il microcircuito VIC II del tuo COMMODORE 64 effettua tutto il lavoro di creare e tenere traccia dei caratteri e delle grafiche, creare i colori e di muovere qua e là gli sprite. Tutto quello che devi fare è passare al computer i seguente tre dettagli sullo sprite:

Cosa dovrà essere
Di che colore sarà
Dove dovrà apparire

Il microcircuito VIC II comprende 46 registeri e controlla fino a 8 sprite alla volta. Tu puoi progettare, creare e muovere il tuo sprite inserendo (POKE) il giusto valore decimale nella particolare locazione di memoria.

7.3. DISEGNO DI UNO SPRITE

Uno sprite è formato da 504 punti, disposti in una griglia di 24 punti in larghezza per 21 punti in altezza. Come accennato prima, puoi utilizzare fino a 8 sprite alla volta, numerati da 0 a 7. Ogni punto dello sprite corrisponde ad un bit. Per disegnare uno sprite, basta disegnare sulla griglia i bit che formano il disegno. Ogni linea della griglia contiene 24 bit (tre byte). Ogni sprite occupa 64 byte in memoria, cioè 21 moltiplicato per 3 più un byte di riserva. (Per i più tecnicamente disposti, il numero 64 è un numero con il quale il circuito VIC II lavora molto più facilmente perché è una potenza di 2 e, quindi, più facile da moltiplicare che non i 63 byte del conto effettivo).

Poiché puoi visualizzare uno sprite all'interno di una griglia, il processo di disegno è enormemente semplificato. Supponiamo che tu voglia creare un aerostato e farlo ondeggiare per lo schermo. La griglia più avanti mostra la sua sagoma. Puoi configurare la tua propria griglia preferibilmente utilizzando carta quadrettata o, meglio ancora, millimetrata. Traccia una griglia alta 21 quadretti e larga 24. Dividi i 24 quadretti con 3 settori (colonne) di 8 spazi ciascuno. Il passo successivo è convertire il disegno grafico in dati utilizzabili dal computer. Numera gli 8 quadretti in ciascuno dei tre settori 128, 64, 32, 16, 8, 4, 2 e 1. Questi valori sono equivalenti a 2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1 e 2^0. Numera i quadretti, del lato sinistro della pagina, da 1 a 21 per ogni fila. Adesso completa la griglia con uno disegno qualsiasi, oppure utilizza l'aerostato che abbiamo disegnato noi. È più facile per prima cosa contornare la sagoma per poi completarla. Pensa ai quadretti che hai riempito come fossero bit ACCESI, e metti un 1 per ogni quadretto pieno. Pensa ai quadretti che non sono riempiti come bit SPENTI e dagli un valore di zero.
Adesso guarda la fila 1 e pensa ad ogni settore di 8 quadretti come fosse un byte. Converti ogni settore di 8 bit in un valore decimale. Volendo potresti utilizzare anche il tuo programma di 'Conversione da Binario a Decimale". Adesso converti ognuna delle 21 file in 3 valori decimali, ottenendo in tutto 63 valori.

```
            SERIE | SERIE | SERIE
              1   |   2   |   3
                  |       |
        1        |1       |1
        2631     |2631    |2631
        842684218426842184268421
       +------------------------+
     1 |.........#######........|
     2 |.......###########......|
     3 |......#############.....|
     4 |......#####...#####.....|
     5 |.....#####.###..####....|
     6 |.....#####.###.#####....|
     7 |.....#####.###..####....|
     8 |......#####...#####.....|
     9 |......#############.....|
 R  10 |......#############.....|
 I  11 |......#.#########.#.....|
 G  12 |.......#.#######.#......|
 A  13 |.......#..#####..#......|
    14 |........#..###..#.......|
    15 |........#..###..#.......|
    16 |.........#..#..#........|
    17 |.........#..#..#........|
    18 |..........#####.........|
    19 |..........#####.........|
    20 |..........#####.........|
    21 |...........###..........|
       +------------------------+
                 1    1    2   2
        1   5    0    5    0   4

               COLONNA
```

Esaminando il disegno dell'aerostato potremo vedere che la prima serie di 8 quadretti, sulla prima fila, sono tutti vuoti, per cui i bit saranno tutti SPENTI dandogli un valore di zero. La serie centrale sulla fila 1 è simile a questa:

```
00000000  01111111  00000000
```

Anche la terza serie di 8 quadretti, sempre sulla prima fila, contiene solo vuoti per cui anche questa sarà uguale a zero. Per cui i dati della prima linea saranno:

```
DATA 0, 127, 0
```

La terza serie di punti che formano la fila due sono calcolati in questo modo:

```
            +-----+-----+-----+-----+-----+-----+-----+-----+
 Series 1:  |  0  |  0  |  0  |  0  |  0  |  0  |  0  |  1  |
            +-----+-----+-----+-----+-----+-----+-----+-----+
                                                         1  =  1

            +-----+-----+-----+-----+-----+-----+-----+-----+
 Series 2:  |  1  |  1  |  1  |  1  |  1  |  1  |  1  |  1  |
            +-----+-----+-----+-----+-----+-----+-----+-----+
               ^     ^     ^     ^     ^     ^     ^     ^
               |     |     |     |     |     |     |     |
             128 +  64 +  32 +  16 +  8  +  4  +  2  +  1  =  255

            +-----+-----+-----+-----+-----+-----+-----+-----+
 Series 3:  |  1  |  1  |  0  |  0  |  0  |  0  |  0  |  0  |
            +-----+-----+-----+-----+-----+-----+-----+-----+
               ^     ^
               |     |
             128 +  64                                      =  192
```

I dati della seconda fila saranno:

```
DATA 1, 255, 192
```

Usa questo sistema per convertire le tre serie di 8 quadretti per ciascuna riga rimanente. Una volta completati questi calcoli, sarai pronto a scrivere un programma BASIC per utilizzare l'aerostato (o qualsiasi altra forma) poiché adesso lo sprite è stato convertito in valori che il tuo computer può capire. Per dimostrare l'impiego degli sprite, immetti questo programma:

```
  1 REM SU, SU, ED OLTRE
  5 PRINT "(CLR/HOME)"
 10 V = 53248 : REM INIZIO DEL MICROCIRCUITO DI VISUALIZZAZIONE
 11 POKE V + 21,4 : REM ABILITA LO SPRITE 2
 12 POKE 2042,13 : REM DATI DELLO SPRITE 2 DAL BLOCCO 13
 20 FOR N = 0 TO 62 : READ Q : POKE 832+N,Q : NEXT
 30 FOR X = 0 TO 200
 40 POKE V + 4,X : REM AGGIORNA COORDINATE X
 50 POKE V + 5,X : REM AGGIORNA COORDINATE Y
 60 NEXT X
70 GOTO 30
200 DATA 0,127,0,1,255,192,3,255,224,3,231,224
210 DATA 7,217,240,7,223,240,7,217,240,3,231,224
220 DATA 3,255,224,3,255,224,2,255,160,1,127,64
230 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
240 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

Riceve le sue informazioni da DATA

Informzioni lette da Q

Se hai immesso ogni cosa correttamente, quando digiti "RUN" e premi <RETURN>, vedrai il tuo aerostato veleggiare attraverso lo schermo. Probabilmente, in questa fase, non capirai il significato della maggior parte del programma ma, via via che spiegheremo ogni fase per la gestione degli sprite, utilizzeremo il programma per spiegarne tutte le caratteristiche.

I numeri di riga dalla 200 alla 240 si riferiscono alle definizione del tuo aerostato e contengono 21 serie di tre valori, cioè una serie per ogni fila della tua tabella del disegno.

7.4. PUNTATORI AGLI SPRITE

Il puntatore allo sprite indica dove si trova il tuo sprite in memoria. I puntatori sono memorizzati in 8 byte, a partire dalla locazione 2040 alla 2047 compresa. La normale locazione del puntatore allo sprite 0 (il primo sprite) è la 2040; la locazione per il puntatore allo sprite 1 è la 2041; e così via fino alla locazione 2047, la locazione del puntatore allo sprite 7 (l'ultimo). Ciascun puntatore allo sprite può contenere un valore tra 0 e 255 che, moltiplicato per 64, corrisponde all'indirizzo di partenza dei tuoi dati dello sprite. Poiché ogni sprite usa 64 byte, il puntatore allo sprite può contenere un numero di blocco da qualche parte entro il primo blocco da 16Kb di memoria, accessibile dal microcircuito VIC II, cioè 256*64. È anche possibile utilizzare altri blocchi da 16Kb. Ulteriori dettagli si possono trovare nella COMMODORE 64 Programmer's Reference Guide.

NOTA: È sempre consigliabile memorizzare i dati del primo sprite nel blocco 255 e poi memorizzare i dati, di eventuali altri sprite, nei successivi blocchi disponibili, verso il basso. In questo modo si impedirà che i tuoi dati dello sprite interferiscano con il programma BASIC. Se rilevi che i dati dello sprite stiano scrivendo sopra la fine del tuo programma BASIC, devi memorizzare i tuoi dati nel successivo blocco di memoria da 16Kb disponibile o spostare il programma BASIC sopra i dati dello sprite. Anche in questo caso i dettagli per come fare questo si possono trovare nella COMMODORE 64 Programmer's Reference Guide.

Nel programma sopra, la riga:

```
V = 53248
```

assegna alla variabile V il valore dell'indirizzo di partenza del VIC II. Più avanti, nel programma, puoi aggiungere il numero del registro all'indirizzo contenuto in V. Per esempio, la riga 11:

```
POKE V + 21,4
```

si riferisce al numero 21 del registro. La riga 12 dello stesso programma:

```
POKE 2042,255
```

inserisce nel blocco 255 i dati dello sprite 2.

7.5. ATTIVAZIONE DEGLI SPRITE

Prima di vedere ed utilizzare i tuoi sprite devi prima attivarli e, per farlo, devi utilizzare il registro SPRITE ENABLE, il registro 21. Come accennato sopra, la riga 11 del programma attiva lo sprite 2, e questo viene fatto inserendo il valore 4 nel registro. Ma perché il 4? Perché 4 non è altro che il 2 elevato alla potenza dello sprite che inizializzi, cioè lo sprite 2. Se non hai capito il motivo per cui noi prendiamo questo valore, ritorna alla struttura di un byte nel capitolo precedente. Se tu volessi attivare due sprite, basterà sommare insieme i due valori decimali. Per esempio, per attivare gli sprite 2 e 3 si sommerà 4 ed 8 (2^2 + 2^3). L'istruzione allora sarà:

```
POKE V + 21,12
```

> SUGGERIMENTO:
> Un modo più facile di attivare un dato sprite è di utilizzare un semplice calcolo che imposti il bit necessario nel registro SPRITE ENABLE. Nell'istruzione di programma qui sotto, SN rappresenta il numero dello sprite (0-7) che vuoi attivare.
>
> ```
> POKE V+21, PEEK(V+21) OR (2^SN)
> ```

7.6. COLORI DEGLI SPRITE

Uno sprite può avere qualsiasi colore dei 16 disponibili sul tuo COMMODORE 64. I colori sono numerati da 0 a 15. Il capitolo 6 e l'Appendice G trattano i colori ed i loro codici. Come puoi vedere dalla Mappa dei Registri del VIC II, ogni sprite possiede il suo proprio registro del colore. I numeri di registro da 39 al 46 vengono utilizzati per questo scopo. Il registro 39 contiene il colore per lo sprite 0, il registro 40 per lo sprite 1, e così via fino al registro 46 che contiene il colore per lo sprite 7. Quando osservi il tuo sprite sullo schermo, i punti vengono visualizzati nel colore contenuto nel registro del colore. Il resto dello sprite è trasparente e mostra qualsiasi colore che si trovi dietro allo sprite.
Se vuoi modificare il colore dello sprite 2 in verde chiaro (numero di codice 13), basta inserire (POKE) il codice del colore nel registro dello sprite che ne regola il colore, come segue:

```
POKE V + 41,13
```

7.7. POSIZIONAMENTO DEGLI SPRITE

Adesso che hai creato uno sprite, vorrai visualizzarlo e spostarlo per lo schermo. Per far questo il tuo COMMODORE 64 usa tre registri di posizione:

a) Registro per la Posizione X dello Sprite
b) Registro per la Posizione Y dello Sprite
c) Registro per la Posizione X Più Significativa

I Registri di Posizione X e Y lavorano insieme per indicare dove apparirà il tuo sprite sullo schermo. Il Registro di Posizione X determina la posizione dello sprite nella direzione orizzontale, mentre il Registro di Posizione Y determina la posizione dello sprite nella direzione verticale. Sulla mappa dei registri del VIC II si nota che i registri da 0 a 15 sono utilizzati per le coordinate X ed Y. I registri sono disposti a coppie come segue:

Il Registro 0 contiene la coordinata X dello sprite 0
Il Registro 1 contiene la coordinata Y dello sprite 0
Il Registro 2 contiene la coordinata X dello sprite 1
Il Registro 3 contiene la coordinata Y dello sprite 1

Questo schema viene ripetuto con i Registri 14 e 15 che contengono le coordinate X ed Y dello sprite 7. C'è un ulteriore registro (il 16) del quale parleremo più avanti.
Puoi posizionare il tuo sprite semplicemente inserendo (POKE) i valori nei registri adatti. Ovviamente per posizionare lo sprite avrai bisogno di entrambe le coordinate X ed Y. Calcola le posizioni a partire dall'ALTO a SINISTRA dell'area del tuo sprite. Non importa quanti punti hai riempito nello schema 24 x 21 per il disegno dello sprite, la posizione sarà sempre calcolata dall'angolo superiore sinistro.
Se esamini ancora il programma dell'aerostato, potrai vedere che i numeri di riga 30-70 utilizzano un ciclo FOR...NEXT per muovere l'aerostato diagonalmente, da sinistra a destra. Queste istruzioni incrementano i valori dello coordinate X e Y inserendo (POKE) le posizioni nei registri 4 e 5, i registri dello sprite 2, finché entrambi i valori non arrivano a 200. In seguito la riga 70 esegue il programma un'altra volta.
Una volta eseguito il programma, hai potuto notare che l'aerostato non si porta molto lontanto verso il lato destro dello schermo. Il posizionamento nella direzione orizzontale è difficile poiché hai bisogno di 320 locazioni e per questo ti occorre un ulteriore bit che, allora, ti darà fino a 512 posizioni. Se non capisci in che modo noi arriviamo a questo numero, pensa ad un bit in più da aggiungere a sinistra di un byte, il che equivarrà ad un 2 elevato alla potenza di 8. I bit in più di tutti gli sprite sono contenuti nel Registro del Bit Più Significativi (MSB), il registro 16. I bit da 0 a 7 di questo registro corrispondono rispettivamente agli sprite da 0 a 7. Se non usi posizioni maggiori di 255, il rispettivo bit di posizione extra deve essere disattivato, cioè deve contenere un valore di zero.
Ecco come lavora l'MSB: dopo che hai spostato lo sprite sulla posizione 255 della X, inserisci (POKE) nel registro 16 il valore decimale dello sprite. Per esempio: per muovere lo sprite 6 sulle posizioni orizzontali dalla 256 alla 320, usa questa istruzione:

```
POKE V + 16,64
```

Quindi usa un ciclo per muovere lo sprite 6 di 64 spazi, dalla posizione 256 alla 320:

FOR X = 0 TO 63 : POKE V + 12,X : NEXT

Il seguente programma revisiona il programma originale dell'aerostato in modo che lo sprite 2 possa muoversi per tutto il tragitto dello schermo:

```
 10 V = 53248 : POKE V + 21,4 : POKE 2042,13

 20 FOR N = 0 TO 62 : READ Q : POKE 832 + N,Q : NEXT
 25 POKE V + 5,100
 30 FOR X = 0 TO 255
 40 POKE V + 4,X
 50 NEXT
 60 POKE V + 16,4
 70 FOR X = 0 TO 63
 80 POKE V + 4,X
 90 NEXT
100 POKE V + 16,0
110 GOTO 30
```

La riga 60 imposta il bit più significativo dello sprite 2.
Le righe dalla 70 alla 90 contengono il ciclo per far muovere lo sprite 2 attraverso le locazioni di schermo dalla 256 alla 320.
La riga 100 disattiva l'MSB in modo che lo sprite 2 possa ritornare al margine sinistro dello schermo. In altre parole, quando l'MSB è ACCESO (1), lo sprite può muoversi solo nelle locazioni dalla 256 alla 320. Prima di far muovere lo sprite nelle locazioni dalla 0 alla 255, devi disattivare (0) l'MSB.
Da notare che il programma che abbiamo utilizzato per attivare i singoli sprite può essere usato anche per impostare un ben preciso MSB. L'istruzione complementare che disattiverà uno specifico bit è:

POKE V+21, PEEK(V+21) AND (255-2^SN)

dove SN è il numero dello sprite che vuoi muovere.

7.8. SPRITE ESPANSI

È possibile aumentare la dimensione di ciascun punto dello sprite in modo da raddoppiarlo in larghezza, in altezza o in entrami le direzioni. Per questo, ci sono due registri EXPAND:

Il Registro 23 raddoppia la larghezza dello sprite
Il Registro 29 raddoppia l'altezza dello sprite

Il sistema per espandere gli sprite è uguale a quello usato durante la loro abilitazione; per esempio, per espandere uno specifico sprite solo nella direzione X, usa la seguente istruzione:

POKE V+23, PEEK(V+23) OR 2^SN

dove SN è il numero dello sprite che vuoi espandere.
Lo stesso discorso vale per raddoppiare l'altezza di uno sprite, tranne che questa volta utilizzi V+29.
Prova ad aggiungere la seguente riga al programma originale dell'aerostato:

```
25 POKE V + 23,4 : POKE V + 29,4 : REM ESPANDE LO SPRITE
```

Quando digiti "RUN", adesso l'aerostato è raddoppiato in dimensione. Questo perché hai inserito (POKE) il valore decimale dello sprite 2 (2^2) nel registro 23 che raddoppia l'altezza dell'aerostato, e nel registro 29 che ne raddoppia la larghezza.

7.9. CREAZIONE DI PIÙ DI UNO SPRITE

La creare e memorizzare di più sprite è una semplice operazione. Le istruzioni su come farlo sono state date in precedenza su questo capitolo. Per aggiungere lo sprite 3 sul tuo schermo, aggiungi le seguenti righe al programma originale dell'aerostato:

```
11 POKE V + 21,12
12 POKE 2042,13 : POKE 2043,13
30 FOR X = 1 TO 190
45 POKE V + 6,X
55 POKE V + 7,190 - X
```

La riga 11 attiva gli sprite 2 e 3 inserendo la somma dei loro valori decimali (4 e 8) nel registro SPRITE ENABLE (21).
La riga 12 dice al computer di cercare i dati degli sprite nel blocco 255 della memoria del VIC II. Ricorda che 2042 è il puntatore allo sprite 2 e 2043 è il puntatore allo 3.
Le righe 45 e 55 muovono lo sprite 3 per lo schermo cambiando i valori ai registri per le coordinate X ed Y di quello sprite (V+6 e V+7).
Quando esegui il programma, vedrai due aerostati che si muovono per lo schermo perché noi abbiamo inserito lo stesso indirizzo in entrambi i puntatori agli sprite.
Le righe successive inseriscono sullo schermo anche lo sprite 4:

```
11 POKE V + 21,28
12 POKE 2042,13 : POKE 2043,13 : POKE 2044,13
25 POKE V + 23,12: POKE V + 29,12
48 POKE V + 8,X
58 POKE V + 9,100
```

La riga 11 attiva gli sprite 2, 3 e 4 inserendo la somma dei lori valori decimali (4, 8 e 16) nel registro SPRITE ENABLE (21).
La riga 12 dice al computer di cercare i dati di tutti e tre gli sprite nel blocco 255 della memoria.
La riga 25 raddoppia la dimensione degli sprite 2 e 3 inserendo la somma dei loro valori nel registro che controlla l'espansione in altezza ed in larghezza (23 e 29).
La riga 48 muove lo sprite 4 a metà strada lungo l'asse X (orizzontale).
La riga 58 posiziona lo sprite 4 a metà strada, in basso, alla locazione 100.
In precedenza, nel programma è stata cambiata la coordinata Y con l'uso di un ciclo FOR...NEXT (guarda la riga 50 del programma originale). Adesso, però, il valore della coordinata Y dello sprite 4 (V+ 9) rimane la stessa per tutto il programma, per cui lo sprite 4 può muoversi solo orizzontalmente.

7.10. PRIORITÀ DEGLI SPRITE

Se usi più di uno sprite, sullo schermo potresti volere far passare uno sprite davanti agli altri. La priorità sprite-sprite è già presente dato che gli sprites con il numero più basso hanno la precedenza più alta, cioè lo sprite 0 è quello che ha la priorità più alta, poi lo sprite 1, lo sprite 2, ecc. fino ad arrivare allo sprite 7 che ha la priorità più bassa di tutti. Gli sprite con la priorità più alta passano davanti agli sprite con la priorità più bassa. La priorità sprite-fondo viene controllata dal registro SPRITE BACKGROUND PRIORITY, il registro 27. I bit 0-7 di questo registro corrispondono agli sprite 0-7 e, di solito, sono impostati su OFF (a zero), il che significa che gli sprite hanno la priorità più alta rispetto al fondo, cioè loro passano davanti a qualsiasi dato presente sullo schermo. Se vuoi scambiare questa priorità di un qualsiasi sprite, devi attivarne il relativo bit. Per esempio, questa istruzione:

POKE V + 27,8

fa apparire lo sprite 3 dietro ad eventuali caratteri che si trovano sullo schermo.

7.11. DISATTIVAZIONE DEGLI SPRITE

Puoi far scomparire uno sprite impostando su OFF (0) il relativo bit del registro SPRITE ENABLE (21). Lo puoi fare con la seguente istruzione:

POKE V+21, PEEK(V+21) AND (255-2^SN)

dove SN è il numero dello sprite che vuoi disattivare.

Operatori Booleani

L'istruzione dell'esempio precedente usa quello che è conosciuto come OPERATORE LOGICO, qualche volta noto come OPERATORE BOOLEANO. In quell'esempio l'AND era l'operatore logico e si utilizza per modificare il primo di due elementi nell'istruzione, cioè il registro 21. L'AND confronta logicamente ciascun rispettivo bit, del risultato dell'istruzione PEEK, in base alle seguenti regole:

1 AND 1 = 1
0 AND 1 = 0
1 AND 0 = 0
0 AND 0 = 0

Questa sopra è conosciuta come TAVOLA DELLA VERITÀ dell'AND. Come puoi vedere, il bit da confrontare viene impostato a zero tranne quando entrambi i bit del confronto contengono 1. Puoi applicare l'istruzione sopra per azzerare i bit in qualsiasi registro di qualsiasi sprite. Per esempio, avresti potuto utilizzare la stessa istruzione, per ridurre la dimensione di uno sprite, semplicemente sostituendo sia il registro 23 che il 29.
Andiamo a vedere cosa succede quando disattiviamo lo sprite 3. Prima che l'istruzione venga eseguita, il registro 21 contiene 00001000. Il risultato dell'espressione dopo l'istruzione PEEK sarà: 255 - 2^3 (8) = 247 oppure 11110111, cioè, dopo aver eseguito l'istruzione il registro 21 conterrà zero. Se in quel momento fossero stati attivi altri sprite, sarebbero rimasti nella stessa condizione poiché entrambi i bit avrebbero contenuto l'1.

L'altro operatore booleano che abbiamo utilizzato è l'OR. La tavola della verità di questo operatore è come segue:

1 OR 1 = 1
0 OR 1 = 1
1 OR 0 = 1
0 OR 0 = 0

Se uno dei due bit è impostato (1), sarà impostato anche il corrispondente risultato.

Questo capitolo ha fatto un'introduzione agli sprite. Prova a sperimentare da te disegnando ed animando i tuoi sprite. Ulteriori dettagli sulla gestione degli sprite si possono trovare nella COMMODORE 64 Programmer's Reference Guide.

8. CREAZIONE DI SUONI E MUSICA

Il tuo computer COMMODORE 64 è equipaggiato con uno tra i più sofisticati sintetizzatori elettronici musicali che si possono trovare su computer. Questo c apitolo è una introduzione all'utilizzo del microcircuito per il suono del tuo computer, il SID. Le caratteristiche principali fornite dal SID sono:

a) Controllo del volume
b) Voci multiple
c) Forma dell'onda
d) Frequenza
e) Generatore di inviluppo (attacco, decadenza, sostegno, rilascio)

8.1. Il circuito integrato del SID

Il microcircuito SID (Dispositivo di Interfaccia Audio) contiene 29 registri ad 8 bit, numerati da 0 a 28, ognuno dei quali è responsabile di un certo componente per la generazione audio. In questo capitolo verrai interessato solo ai primi 25 registri che sono riposti tra le locazioni 54272 e 54296 incluse.
Ecco un riassunto del mappa dei registri del SID:

REGISTRO N°	DESCRIZIONE
0-6	VOCE 1
7-13	VOCE 2
14-20	VOCE 3
21	BASSA FREQUENZA
22	ALTA FREQUENZA
24	CONTROLLO VOLUME E FILTRI

Prima di andare avanti nella discussione sul modo in cui vengono creati i suoni, immetti il programma che segue ed eseguilo (RUN). Questo dimostrerà solo una piccola parte di ciò che si può ottenere dal SID.

```
PROGRAMMA ESEMPIO 1:

  5 S=54272
 10 FOR L=S TO S+24:POKE L,0:NEXT: REM LIBERA IL CIRCUITO AUDIO
 20 POKE S+5,9:POKE S+6,0
 30 POKE S+24,15: REM IMPOSTA IL LIVELLO MAZZIMO DEL VOLUME
 40 READ HF,LF,DR
 50 IF HF<0 THEN END
 60 POKE S+1,HF:POKE S,LF
 70 POKE S+4,33
 80 FOR T=1 TO DR:NEXT
 90 POKE S+4,32:FOR T=1 TO 50:NEXT
100 GOTO 40
110 DATA 25,177,250,28,214,250
120 DATA 25,177,250,25,177,250
130 DATA 25,177,125,28,214,125
140 DATA 32,94,750,25,177,250
150 DATA 28,214,250,19,63,250
160 DATA 19,63,250,19,63,250
170 DATA 21,154,63,24,63,63
180 DATA 25,177,250,24,63,125
190 DATA 19,63,250,-1,-1,-1
```

La riga 5 memorizza in S la locazione di partenza del SID. Si può accedere a tutti gli altri registri semplicemente aggiungendo i loro numeri ad S.

CONTROLLO VOLUME

Il tuo COMMODORE 64 ha 16 livelli di volume, numerati da 0 (spento) a 15 (volume massimo). Il registro 24 del SID controlla il livello del volume. Per impostare il volume basta inserire (POKE) in questo registro il valore che vuoi. La riga 30 del programma d'esempio imposta il volume sul livello massimo (15).
Di solito dovrai impostare il volume solo all'inizio del tuo programma. Da notare che il livello del volume stabilisce l'uscita per tutte le tre voci del tuo 64.

VOCI

Il tuo COMMODORE 64 ha tre voci che possono essere riprodotte separatamente o simultaneamente. Qui sotto viene mostrata la mappa dei registri di ciascuna voce:

NUMERI DEL REGISTRO

VOCE 1	VOCE 2	VOCE 3	DESCRIZIONE
0	7	14	VALORI PER BASSA FREQUENZA
1	8	15	VALORI PER ALTA FREQUENZA
2	9	16	AMPIEZZA BASSA DELL'IMPULSO
3	10	17	AMPIEZZA ALTA DELL'IMPULSO
4	11	18	REGISTRO DI CONTROLLO
5	12	19	SETTAGGI ATTACCO/DECADIMENTO
6	13	20	SETTAGGI SOSTEGNO/RILASCIO

FREQUENZA

Il suono viene creato dal movimento dell'aria. Immagina di gettare una pietra in un stagno e stare a vedere le onde che si irradiano verso l'esterno. Quando questo tipo di onde vengono create nell'aria, noi udiamo un suono. Ogni suono prodotto dal tuo 64 è formato da un valore di frequenza alta e bassa. Ognuna delle tre voci del 64 ha due registri in cui sono contenuti i valori della frequenza. I due valori in ciascuna voce sono combinati in modo da formare il valore della frequenza in una struttura a 16 bit. Qui sotto viene mostrato un diagramma delle locazioni di memoria dei registri contenenti la frequenza alta e bassa di ciascuna voce:

VOCE	FREQUENZA	NUMERO POKE
1	ALTA	54273
1	BASSA	54272
2	ALTA	54280
2	BASSA	54279
3	ALTA	54287
3	BASSA	54286

Per riprodurre una nota musicale o un suono, devi inserire (POKE) il valore della frequenza alta del suono nella locazione della frequenza alta della voce che vuoi, ed inserire il valore della frequenza bassa della nota nella locazione della frequenza bassa della voce. La riga 60 del programma d'esempio inserisce le frequenze alte e basse, attraverso le istruzioni DATA, dentro i registri 1 e 0 rispettivamente. In questo modo si imposta la frequenza per la voce 1.

CREAZIONE DI ALTRE FREQUENZE

Per creare una frequenza diversa da quelle elencate nella tabella delle note, utilizza la seguente formula:

F = FYOUT / .06097

dove FYOUT è la frequenza che richiedi.
Per creare i valori di frequenza alta e bassa delle note, prima di tutto devi rendere intero F, cioè eliminare tutti i numeri alla destra del punto decimale.
Adesso utilizza questa formula per calcolare la locazione della frequenza alta:

HI = INT (F / 256)

e la formula che segue per avere la locazione della frequenza bassa:

LO = F - (256 * HI)

Quindi, per ottenere la nota, basterà inserire (POKE) il valore LO nel registro della frequenza bassa ed il valore HI nel registro della frequenza alta della voce dalla quale vuoi far uscire il suono.

FORME D'ONDA

Il tipo di forma d'onda che scegli stabilisce il timbro o la qualità del suono prodotto.
Nel tuo 64 ci sono quattro tipi di forme d'onda:

TRIANGOLO. Questa forma d'onda comprende poche armoniche ed un suono caldo simile al flauto. La forma dell'onda triangolare è simile a questa:

```
                    /\
                   /  \
                  /    \
............../............\..............
   \        /              \        /
    \      /                \      /
     \/                      \/
```

DENTE DI SEGA. Questa forma d'onda contiene tutte le armoniche ed ha la qualità brillante degli strumenti d'ottone. Ecco a cosa assomiglia l'onda a dente di sega:

```
         +             +             +
        /|            /|            /|
       / |           / |           / |
      /  |          /  |          /  |
.........|..... /......|....../......|........
         |    /        |    /        |    /
         |  /          |  /          |  /
         |/            |/            |/
         +             +             +
```

ONDA AD IMPULSO VARIABILE. Questa forma d'onda comprende onde rettangolari variabili. Modificando l'ampiezza dell'impulso i suoni variano da un rumore chiaro e cupo, ad un impulso nasale, acuto. Ecco a cosa assomiglia:

```
  +-----------+   +-----------+   +-----------+   |
  |           |   |           |   |           |   |
  |           |   |           |   |           |   |
  |           |   |           |   |           |   |
..|...........|...|...........|...|...........|...|..
  |           |   |           |   |           |   |
  |           |   |<--------->|   |           |   |
  |           |   | LARGHEZZA |   |           |   |
  |           +---+  IMPULSO  +---+           +---+
```

RUMORE BIANCO. Questa forma d'onda viene utilizzata principalmente per gli effetti sonori (per es. esplosioni, colpi d'arma da fuoco, risacca) e varia da un basso brontolio fino ad un sibilo. È simile a questA:

```
                  .    .         .
      .      . .              .    .
       .   .      .         .
 . . . . . . . . . . . . . . . . . . . . . . . . . . .
          . . .           .
                   .   .  .   .
         .                        .  .
```

Le forme d'onda di ciascuna voce sono contenute in tre registri di controllo, numerati 4, 11 e 18. Le parti componenti del registro di controllo di ciascuna voce sono i seguenti:

BIT N°	DESCRIZIONE
0	PORTA
1-3	NON USATO
4	FORMA D'ONDA TRIANGOLO
5	FORMA D'ONDA DENTE DI SEGA
6	FORMA D'ONDA AD IMPULSO
7	FORMA D'ONDA RUMORE

Il bit di PORTA controlla il Generatore d'inviluppo. Quando questo bit è impostato ad 1, attiva il Generatore d'inviluppo ed inizia il ciclo ATTACCO/DECADIMENTO/SOSTEGNO. Quando il bit è resettato, inizia il ciclo di RILASCIO. Impostando ad 1 i bit 4, 5 o 6 si seleziona quella particolare forma d'onda.
La riga 70 del programma imposta l'uscita suono della voce 1 utilizzando una forma d'onda a dente di sega. Questa riga imposta anche il bit di PORTA. Questi bit li puoi anche impostare in combinazioni, cioè Impulso e Dente di sega, ma si produrranno suoni piuttosto strani!

IL GENERATORE DI INVILUPPO

Il volume di un tono musicale cambia dal momento in cui cominci a sentirlo fino a quando non scompare e non puoi più sentrlo. Quando una nota viene prodotta la prima volta, aumenta da un volume a zero fino al suo volume di picco. L'andamento in cui avviene si chiama ATTACCO. In seguito perde il picco e si assesta a un livello medio di volume. L'andamento in cui avviene questo assestamento si chiama DECADIMENTO. Una volta raggiunto il volume medio, tutto il tempo in cui rimane la nota si chiama livello di SOSTEGNO. Quando infine la nota smette di suonare, perde il livello di SOSTEGNO fino al volume a zero. L'andamento in cui cade si chiama RILASCIO. Il disegno che segue mostra le quattro fasi di una nota:

```
                                +
                               /.\
                              / .  \
                             /  .    \
 LIVELLO DI SOSTEGNO . . / . . . . +-----------+
                           /    .     .           . \
                          /     .     .           .   \
                         /      .     .           .     \
                        .       .     .           .       .
                        .       .     .           .       .
                        .   A   .  D  .     S     .   R   .
```

NOTA: **A**TTACCO, **D**ECADIMENTO e **R**ILASCIO sono ANDAMENTI. Il **S**OSTEGNO è un LIVELLO.

Ognuno dei cicli sopra danno certe qualità e limitazioni alla forma, o INVILUPPO, di un suono. Questi limiti vengono collettivamente chiamati parametri. Detti parametri di ATTACCO/DECADIMENTO/SOSTEGNO/RILASCIO vengono collettivamente chiamati ADSR.
Come parametri ADSR per ognuna delle tre voci, ci sono due registri. Questi sono il 5 ed il 6 per la voce 1, il 12 ed il 13 per la voce 2, ed il 19 ed il 20 per la voce 3. I parametri di ATTACCO e DECADIMENTO condividono i primi di ogni coppia di registri (5, 12, 19) mentre i parametri SOSTEGNO e RILASCIO utilizzano i registri 6, 13 e 20.
Si utilizzano questi accoppiamenti perché le impostazioni richiedono solo 4 bit o metà byte. Questa quantità di carica è chiamata un NYBBLE. I primi quattro bit di un byte rappresenta il NYBBLE ALTO mentre gli ultimi quattro bit rappresentano NYBBLE BASSO. Le impostazioni per l'ATTACCO delle tre voci sono contenute nei nybble alti dei registri 5, 12 e 19, mentre le impostazioni di DECADIMENTO sono contenute nei nybble bassi di questi stessi registri. Le impostazioni di SOSTEGNO delle tre voci utilizzano i nybble alti dei registri 6, 13 e 20, mentre le impostazioni di RILASCIO utilizzano i nybble bassi degli stessi registri. Prima di inserire (POKE) un valore qualsiasi nei registri ADSR, devi prima riunire i nybble alti e bassi aggiungendoli insieme. Per esempio, gli andamenti d'ATTACCO occupano i bit 2^7, 2^6, 2^5 e 2^4, per cui i valori saranno 128, 64, 32 e 16. Gli andamenti di DECADIMENTO usano i bit 2^3, 2^2, 2^1 e 2^0, oppure 8, 4, 2 e 1. Supponiamo che tu voglia impostare un valore alto d'ATTACCO (12) ed un valore basso di DECADIMENTO (2). Un modo facile per mettere insieme i due andamenti è di moltiplicare il valore d'ATTACCO per 16 ed aggiungerlo al valore di DECADIMENTO. In questo esempio il valore che ne risulta è 194, cioè 12*16+2. Puoi usare questa formula ogni volta che ti serve per mettere insieme due valori (campo 0-15) in un formato nibble alto/basso.
La riga 20 del programma d'esempio imposta l'andamento ATTACCO/DECADIMENTO. A 0 l'ATTACCO ed a 9 il DECADIMENTO. L'andamento massimo dell'ATTACCO si raggiunge utilizzando un valore a 15 e moltiplicandolo per 16. Puoi incrementare l'andamento del DECADIMENTO unendo tutti i valori del DECADIMENTO, cioè 8+4+2+1=15, il che è l'ANDAMENTO MASSIMO DI DECADIMENTO.

Ecco qui alcune POKE d'esempio per l'ATTACCO/DECADIMENTO:

	VOCE	ATTACCO	DECADIMENTO
POKE 54277,66	1	MEDIO(64)	BASSO(2)
POKE 54284,66	2	MED(64) + BASSO(32)	MEDIO(4)
POKE 54291,15	3	ZERO	MASSIMO (8 + 4 + 2 + 1)
POKE 54284,255	2	MASSIMO (128 + 64 + 32 + 16	MASSIMO + 8 + 4 + 2 + 1)

Qui c'è un programma d'esempio che spiega quello che puoi fare con le impostazioni di ATTACCO/DECADIMENTO:

```
10 FOR L=54272 TO 54296:POKE L,0:NEXT  .. Libera il SID
20 POKE 54296,15  ...................... Imposta il volume al massimo
30 POKE 54277,64  ...................... Imposta ATTACCO/DECADIMENTO
40 POKE 54273,162:POKE 54272,37  ........ Inserisce una nota nella voce 1
50 PRINT"PREMI UN TASTO QUALSIASI" ...... Messaggio su schermo
60 GET K$:IF K$="" THEN 60  ............ Controlla la tastiera
70 POKE 54276,17:FOR T=1 TO 200:NEXT .... Avvia l'onda triangolare
80 POKE 54276,16:FOR T=1 TO 50:NEXT  .... Arresta la nota
90 GOTO 50  ............................ Ripete l'esecuzione
```

Dopo aver eseguito (RUN) il programma alcune volte, prova a cambiare l'impostazione ATTACCO/DECADIMENTO della riga 30:

```
30 POKE 54277,190
```

Adesso riesegui (RUN) il programma e nota la differenza nella nota. Prova altre combinazioni di impostazioni ATTACCO e DECADIMENTO per avere un'idea di come puoi utilizzare i diversi andamenti ATTACCO/DECADIMENTO per creare una varietà di effetti sonori.

Impostazione SOSTEGNO/RILASCIO. Come l'ATTACCO/DECADIMENTO, il SOSTEGNO/RILASCIO condividono un byte. Tuttavia ricorda che questa condivisione non vuol dire che il SOSTEGNO ed il RILASCIO siano simili. Il SOSTEGNO è un LIVELLO, mentre il RILASCIO, l'ATTACCO ed il DECADIMENTO sono ANDAMENTI.
Il SOSTEGNO è una proporzione del volume massimo. Puoi sostenere, o bloccare, le note ed i suoni in qualsiasi livello (16) del volume. Questa tabella ti mostra quali numeri inserire (POKE) per i valori di SOSTEGNO/RILASCIO:

SOSTEGNO ALTO	SOSTEGNO MEDIO	SOSTEGNO BASSO	SOSTEGNO + BASSO	RILASCIO ALTO	RILASCIO MEDIO	RILASCIO BASSO	RILASCIO + BASSO
128	64	32	16	8	4	2	1

NOTA: Puoi incrementare il livello del SOSTEGNO aggiungeNDO insieme tutti i valori di SOSTEGNO: 128 + 64 + 32 + 16 = 240, che è il LIVELLO MASSIMO del SOSTEGNO. Un livello di SOSTEGNO a 128 rappresenta circa il 50% del volume.
Puoi incrementare l'andamento del RILASCIO aggiungendo tutti insieme i valori di RILASCIO: 8 + 4 + 2 + 1 = 15, che è l'ANDAMENTO MASSIMO DI RILASCIO.
Combina il livello di SOSTEGNO e l'andamento di RILASCIO allo stesso modo in cui combini gli andamenti d'ATTACCO e di DECADIMENTO: aggiungi i due valori ed inserisci (POKE) il totale nella locazione di memoria della voce che vuoi.
Per vedere gli effetti della impostazione del livello di sostegno, aggiungi questa riga all'ultimo programma d'esempio:

```
35 POKE 54278,128
```

Adesso esegui (RUN) il programma un'altra volta ed osserva la modifica. Con la riga 35 noi diciamo al computer di sostenere la nota a un LIVELLO ALTO di SOSTEGNO (128). Puoi variare la durata di una nota cambiando il conteggio della riga 70. Ricorda che il livello di sostegno mantiene una nota ad una proporzione di volume via via che la nota perde il suo volume di picco; questa non è la stessa cosa della durata della nota.
Per vedere l'effetto sull'andamento del rilascio, prova a modificare la riga 35 in POKE 54278,89 (SOSTEGNO = 80, RILASCIO = 9).

8.2. PROGRAMMA AUDIO D'ESEMPIO

Questo breve programma sonoro riassume quello che hai imparato fin qui sulla creazione di musica su tuo 64:

```
 5 FOR L=54272 TO 54296:POKE L,0:NEXT
10 POKE 54296,15
20 POKE 54277,190
30 POKE 54278,248
40 POKE 54273,16:POKE 54272,195
50 POKE 54276,17
60 FOR T=1 TO 250:NEXT
70 POKE 54276,16
```

1. Scegli la voce (o le voci) che vuoi utilizzare. Ricorda che ogni voce utilizza locazioni di memoria differenti in cui inserirai (POKE) i valori per la forma d'onda, andamento d'ATTACCO, ecc. Puoi riprodurre 1, 2 o 3 voci insieme, ma questo programma utilizza unicamente la voce 1.

2. Svuota il SID (riga 5)
3. Imposta il VOLUME (riga 10)
4. Imposta gli andamenti ATTACCO/DECADIMENTO: per definire la velocità di aumento di una nota e perdere il suo livello di picco del volume (0-255) (riga 20)
5. Imposta il SOSTEGNO/RILASCIO per definire il livello di mantenimento della nota e l'andamento per rilasciarla (riga 30)
6. Cerca la nota che vuoi eseguire nella TABELLA DELLE NOTE MUSICALI nell'Appendice M ed inserisci i valori di FREQUENZA ALTA e FREQUENZA BASSA di quella nota (ogni nota richiede 2 POKE) (riga 40)
7. Inizializza la FORMA D'ONDA (qui, il TRIANGOLO) (riga 50)
8. Inserisci un ciclo da temporizzare tra le note (noi usiamo 250 per una nota da un quarto) (riga 60)
9. ARRESTA la nota disattivando la forma d'onda scelta (riga 70)

Qui sotto si trova un programma più lungo che dimostra meglio le possibilità di creare musica con il tuo 64:

```
  5 REM SCALA MUSICALE
  7 FOR L=54272 TO 54296:POKE L,0:NEXT  svuota il SID
 10 POKE 54296,15  .................... imposta il VOLUME
 20 POKE 54277,7:POKE 54278,133  ...... imposta l'ADSR
 50 READ A  ........................... legge il primo numero dalla riga 110
 55 IF A=-1 THEN END  ................. termina il ciclo
 60 READ B  ........................... legge il secondo numero
 80 POKE 54273,A:POKE 54272,B  ........ inserisce il primo numero dalla riga 110
                                          come FREQUENZA ALTA ed il secondo
                                          numero come FREQUENZA BASSA
 85 POKE 54276,17  .................... avvia la nota
 90 FOR T=1 TO 250:NEXT:POKE 54276,16   lascia suonare la nota. poi la ferma
 95 FOR T=1 TO 50:NEXT  ............... imposta il tempo di RILASCIO, il tempo
                                          fra le note
100 GOTO 20  .......................... ricomincia il programma
110 DATA 16,195,18,209,21,31,22,96 .... elenca il valore della nota
120 DATA 25,30,28,49,31,165,33,135 .... dal diagramma in Appendice M. Ogni parte
                                          dei numeri = una nota
                                          (16 e 19 = quarta ottava DO)
999 DATA -1  .......................... termina il programma (vedi riga 55)
```

Puoi cambiare in un'onda a dente di sega modificando la riga 85 in POKE 54276,33 e la riga 90 in FOR T = 1 TO 250:NEXT:POKE 54276,32. Cambiando la forma d'onda può cambiare drammaticamente il suono prodotto dal tuo computer. Puoi anche modificare il suono in altri modi. Per esempio, puoi modificare il suono arpicordo, del programma precedente, in un suono simile a quello di un banjo cambiando l'andamento d'ATTACCO/DECADIMENTO di ciascuna nota. Fallo modificando la riga 20 in:

```
20 POKE 54277,3:POKE 54278,0  ........ crea l'effetto banjo
                                        impostando a zero il SOSTEGNO
```

Come dimostra questo programma, il tuo 64 può suonare in una varietà di strumenti musicali.

8.3. RIPRODURRE UNA CANZONE SUL TUO 64

Il programma che segue ti permette di riprodurre una riga da un canzone, "Michael Row Your Boat Ashore". Il programma utilizza la forma d'onda ad IMPULSO, che è un'onda rettangolare a larghezza variabile. Il terzo ed il quarto POKE nella riga 10 definiscono la larghezza dell'impulso di questo brano.
In questo brano noi utilizziamo un calcolo di durata di 125 per un'ottava di nota, 250 per un quarto di nota, 375 per una nota puntata, 500 per metà nota, e 1000 per una nota intera. Quando programmi le

tue canzoni personali, puoi incrementare o diminuire queste quantità per adeguarsi a un particolare tempo o al tuo proprio gusto musicale.

```
  2 FOR L=54272 TO 54296: POKE L,0: NEXT
  5 S=54272
 10 POKE S+24,15: POKE S+5,88: POKE S+3,15: POKE S+2,15: POKE S+6,89
 20 READ H: IF H=-1 THEN END
 30 READ L
 40 READ D
 60 POKE S+1,H: POKE S,L: POKE S+4,65
 80 FOR T=1 TO D: NEXT: POKE S+4,64
 85 FOR T=1 TO 50: NEXT
 90 GOTO 20
100 DATA 33,135,250,42,62,250,50,60,250,42,62,125,50,60,250
105 DATA 56,99,250
110 DATA 50,60,500,0,0,125,42,62,250,50,60,250,56,99
115 DATA 1000,50,60,500
120 DATA -1
```

La riga 2 libera il SID. La La riga 5 assegna ad S la locazione più bassa della memoria del SID. Per tutto il resto del programma noi aggiungiamo a questa variabile il numero del registro del SID. Per esempio, POKE S+24,15 inserisce 15 al registro del volume che è 54296, oppure 54272+24.
La riga 10 inserisce (POKE) i valori in:

1. Il registro del volume: POKE S+24,15
2. Voce 1, andamenti di ATTACCO/DECADIMENTO: POKE S+5,88
3. Largezza dell'impulso: POKE S+3,15 e POKE S+2,15
4. Voce 1, livello di SOSTEGNO/andamento di RILASCIO: POKE S+6,89

La riga 20 legge (READ) il primo numero dall'istruzione DATA. Quando il numero è -1, il programma termina (END) automaticamente. Questa situazione succede quando viene letta l'ultima istruzione DATA (riga 120).
La riga 30 legge (READ) il secondo numero dall'elenco DATA.
La riga 40 legge (READ) il terzo numero dall'elenco DATA.
La riga 60 inserisce (POKE):

1. Il valore di H che è stato assegnato nell'istruzione READ H nella riga 20. Finché non viene letto il -1, questo valore viene assegnato al registro della FREQUENZA ALTA.
2. Il valore di L che è stato assegnato nell'istruzione READ L nella riga 30. Questo valore viene assegnato al registro della FREQUENZA BASSA. Insieme, questi due POKE stabiliscono l'altezza di una nota.
3. Il valore che ATTIVA la forma d'onda dell'impulso variabile della voce 1.

La riga 70 usa un ciclo per impostare la durata della nota da riprodurre. Il valore di D viene assegnato nell'istruzione READ della riga 40. Come puoi vedere, la quantità negli elenchi DATA sono raggruppati a tre per tre: il primo numero (per es., 35) è il valore della frequenza alta di una nota, il secondo numero (per es., 135) è il valore della frequenza bassa della stessa nota, ed il terzo numero (per es., 250) imposta la durata di quella nota (per es., un quarto di nota del DO).
La riga 80 è un ciclo di temporizzazione che determina il tempo di rilascio tra le note.
La riga 90 riporta indietro il programma a leggere (READ) la serie di numeri per un'altra nota.
Le righe dalla 100 alla 120 contengono tutti i DATA per la riga di questa canzone.

8.4. CREAZIONE DI EFFETTI AUDIO

Il SID del tuo 64 ti permette di creare un'ampia varietà di effetti sonori, come un'esplosione durante un gioco, o un cicalino che ti avverte quando fai un errore.
Qui ci sono alcuni suggerimenti per creare effetti sonori:

1. Varia rapidamente tra due note per creare un suono tremolante.
2. Usa gli effetti a più voci per riprodurre più di una voce alla volta, con ciascuna voce controllata indipendenteménte, in modo da avere diversi rumori alla volta. Oppure usa una voce come eco o risposta ad un'altra voce.
3. Usa diverse ampiezze d'impulso per creare suoni diversi.
4. Usa la FORMA D'ONDA del RUMORE per mettere il rumore bianco ed accentuare gli effetti audio tonali, per crea fragori d'esplosione, colpi d'arma da fuoco, passi o allarmi. Quando usi la forma d'onda del rumore con le stesse note musicali con le quali crei la musica, puoi creare diversi tipi di rumore bianco.
5. Combina diverse FREQUENZE ALTE/BASSE in rapida successione su ottave differenti.
6. Prova ad inserire (POKE) le impostazioni extra delle note nell'Appendice M.

Qui ci sono alcuni esempi di programmi per effetti audio. La Guida di Riferimento del Programmatore del Commodore 64 comprende più esempi come pure maggiori informazioni per la creazione di effetti audio.

PIANTO DI BAMBOLA

```
   5 FOR L=54272 TO 54296:POKE L,0:NEXT  Libera il SID
  10 S=54272:POKE S+3,15:POKE S+2,40
  20 POKE S+24,15  ..................... Imposta il volume
  30 POKE S+4,65  ...................... Attiva la forma d'onda impulso nella voce 1
  40 POKE S+5,15  ...................... Imposta l'andamento
ATTACCO/DECADIMENTO
  50 FOR X=200 TO 5 STEP -2  ........... Imposta il ciclo di temporizzazione per il
                                           RILASCIO o il tempo tra le note
  60 POKE S+1,40:POKE S,X:NEXT  ........ Imposta le frequenze alta/bassa
  70 FOR X=150 TO 5 STEP -2  ........... Imposta il ciclo di temporizzazione più veloce
  80 POKE S+1,40:POKE S,X:NEXT  ........ Imposta le frequenze alta/bassa
  90 POKE S+4,0  ....................... Disattiva la forma d'onda impulso
```

SPARO

```
   5 FOR L=54272 TO 54296:POKE L,0:NEXT  Libera il SID
  10 S=54272
  20 FOR X=15 TO 0 STEP -1  ............ Configura il ciclo del volume
  30 POKE S+24,X  ...................... Inserisce X nel registro del volume
  40 POKE S+4,129  ..................... Inizializza la forma d'onda RUMORE
  50 POKE S+5,15  ...................... Imposta l'andamento ATTACCO/DECADIMENTO
  60 POKE S+1,40  ...................... Imposta la frequenza alta
  70 POKE S,200:NEXT  .................. Imposta la frequenza bassa
  80 POKE S+4,128  ..................... Arresta la forma d'onda RUMORE
  90 POKE S+5,0  ....................... Inserisce 0 sull'ATTACCO/DECADIMENTO
  100 GOTO 20  ......................... Ripete il programma
```

La sequenza che inizia nella riga 20 configura la dissolvenza del volume in modo che il rumore del colpo d'arma da fuoco inizi ad un volume alto (15) e diminuisca gradualmente a 0 via via che il ciclo viene eseguito.
Per arrestare questo programma premi il tasto <STOP>.

Come abbiamo detto in precedenza, il modo migliore per imparare un nuovo campo di programmazione è di provare.

8.5. FILTRAGGIO

Qualche volta una certa forma d'onda può non avere affatto il timbro che richiedi. Per esempio: sarebbe difficile immaginare che una delle forme d'onda preimpostate nel SID possa eseguire qualcosa di simile ad una tromba. Per darti un ulteriore controllo sui parametri del suono, il SID è equipaggiato con tre FILTRI.

FILTRO PASSA-ALTO. Questo filtro riduce il livello delle frequenze al di sotto di una ben precisa frequenza di taglio. In pratica fa passare tutte le frequenze alla stessa altezza ed al di sopra della linea di taglio, mentre riduce le frequenze al di sotto di detta linea.

FILTRO PASSA-BASSO. Come suggerisce il suo nome, questo filtro fa passare le frequenze al di sotto della linea di taglio e riduce il livello di quelle al di sopra.

FILTRO PASSA-BANDA. Questo filtro fa passare una stretta banda di frequenze intorno alla linea di taglio e riduce il livello di tutte le altre.

Un ulteriore filtro, chiamato **FILTRO SCARTA PASSO**, può essere sintetizzato combinando i filtri passa-alto e passa-basso. Questo filtro fa passare le frequenze lontane dalla linea di taglio mentre riduce il livello alla frequenza di taglio.

Il registro 24 stabilisce quale tipo di filtro vuoi utilizzare. Ricorda che questo è anche il registro utilizzato per il controllo del volume. Per i filtri si utilizzano i seguenti bit:

BIT N°	USO
4	SELEZIONA IL FILTRO PASSA-BASSO
5	SELEZIONA IL FILTRO PASSA-BANDA
6	SELEZIONA IL FILTRO PASSA-ALTO

Un filtro viene attivato impostando nel registro 24 il bit attinente. Non puoi filtrare tutte le voci contemporaneamente. Il registro 23 stabilisce quali voci devono essere filtrate. I bit sono i seguenti:

BIT N°	USO
7-4	FILTRO DI RISONANZA 0-15
3	FILTRO D'INGRESSO ESTERNO
2	FILTRO PER VOCE 3
1	FILTRO PER VOCE 2
0	FILTRO PER VOCE 1

Quando viene impostato un bit ben preciso, l'uscita di quella voce verrà deviata verso il filtro.
La frequenza di taglio è un numero ad undici bit. Gli otto bit superiori (11-3) sono contenuti nel registro 22 mentre i tre bit più bassi (0-2) sono contenuti nel registro 21. Questo sistema ti offre un campo di valori tra 0 e 2047.
Prova ad aggiungere le seguenti righe al programma d'esempio per filtrare la voce ed ascolta la differenza del suono. Noi utilizzeremo un filtro Passa-Basso che permetterà di ascoltare solo le componenti più basse dei suoni.

```
30 POKE S+24,31: REM TUTTO IL VOLUME PIÙ FILTRO PASSA-BASSO
35 POKE S+23,1: REM SCELTA DEL FILTRO PER LA VOCE 1
37 POKE S+22,128:POKE S+21,7: REM SCELTA DELLA FREQUENZA DI TAGLIO
```

Prova a fare esperimenti con i filtri. Filtrando un suono mentre attraversa le fasi ADSR della sua durata, si possono produrre effetti interessanti. Per ulteriori informazioni su come utilizzare il SID, consulta la Guida di Riferimento del Programmatore del COMMODORE 64.

8.6. COMPOSITORE MUSICALE

La cartuccia MUSIC COMPOSER per il Commodore ti permette di comporre musica sul tuo COMMODORE 64 senza doverti preoccupare del funzionamento del SID. I mezzi che sono forniti ti permettono di immettere righe di programma che comprendono solo particolari caratteri di controllo. Questo metodo ti consente di riprodurre qualunque combinazione di suoni richiesta utilizzando tutte le caratteristiche del SID. Una volta composto il tuo capolavoro, puoi salvarlo su nastro e quindi riprodurlo a tuo piacere. Mentre la musica suona, un pentagramma musicale scorre attraverso lo schermo visualizzando le note via via che vengono eseguite. Questo ti permette di ottenere il massimo dal SID con il minimo sforzo.

9. GESTIONE AVANZATA DEI DATI

9.1. ISTRUZIONI READ E DATA

Fin qui abbiamo mostrato come assegnare valori direttamente alle variabili (A = 2), e come assegnare valori mentre il programma è in esecuzione (INPUT e GET).
Ma spesso troverai che, in un programma, specialmente con una grande quantità di dati, nessuno dei due sistemi soddisfa le tue necessità di assegnazione alle variabili.
Nel Capitolo 7, quando abbiamo introdotto gli sprite, abbiamo usato le istruzioni READ e DATA per assegnare i valori agli sprite. Ecco un breve programma che ti mostra come lavorano insieme queste due istruzioni:

```
10 READ X
20 PRINT "X ADESSO È :"; X
30 GOTO 10
40 DATA 1, 34, 10.5, 16, 234.56

RUN
X ADESSO È : 1
X ADESSO È : 34
X ADESSO È : 10.5
X ADESSO È : 16
X ADESSO È : 234.56
?OUT OF DATA ERROR IN 10
READY.
_
```

La riga 10 legge (READ) un valore dall'istruzione DATA nella riga 40 ed assegna il valore ad X.
La riga 30 dice al computer di ritornare alla riga 10, e qui il READ assegna ad X il valore successivo dell'istruzione DATA. Il ciclo prosegue fino a leggere tutti i valori di DATA.
Quando usi le istruzioni DATA, ci sono alcune regole importanti che devi ricordare:

- Seguire esattamente il formato dell'istruzione Data:

  ```
  40 DATA 1, 34, 10.5, 16, 234.56
  ```

  La virgola separa ciascun elemento.

- Impiego:

  -- numeri interi (per es., 34),
  -- numeri reali (per es., 234.56),
  -- numeri espressi in notazione scientifica (per es., 2.4E+04),
  -- parole (purché tu stia usando una variabile di stringa nell'istruzione READ),

- però nelle istruzioni DATA NON puoi usare:

    -- variabili o
    -- operazioni aritmetiche

    Gli elementi elencati qui sotto, se cerchi di leggerli (READ), vengono considerati stringhe, e li puoi leggere (READ) come tali solo definendo variabili di stringa nell'istruzione READ.

    DATA A, 23/56, 2*5, B + 2

Quando usi un'istruzione READ, puoi prendere valori solo da un'istruzione DATA perché le due istruzioni operano in associazione. Ogni volta che leggi (READ) un valore, il computer sa di spostarsi verso il valore successivo nell'istruzione DATA. In pratica, nel computer c'è un puntatore che tiene traccia della tua posizione nell'istruzione DATA. Dopo aver letto (READ) il primo valore, l'istruzione DATA è simile a questa:

```
40 DATA 1, 34, 10.5, 16, 234.56
                   |
                puntatore
```

Non appena viene assegnato l'ultimo valore del DATA alla variabile nell'istruzione READ ed il computer cerca di eseguire il ciclo un'altra volta, viene visualizzato OUT OF DATA ERROR per il semplice motivo che non ci sono più valori da assegnare, ma al computer non gli è stato detto.
Ecco un esempio che mostra un modo per evitare l'OUT OF DATA ERROR.

```
NEW

10 FOR X = 1 TO 3
15 READ A$
20 PRINT "A$ ADESSO È : "; A$
30 NEXT
40 DATA QUESTO, E’, DIVERTENTE

RUN
A$ ADESSO È : QUESTO
A$ ADESSO È : E’
A$ ADESSO È : DIVERTENTE
READY.
```

Questa volta abbiamo inserito l'istruzione READ dentro un ciclo FOR/NEXT che limita la quantità di letture agli elementi dell'istruzione DATA. Questo metodo va bene, ma implica l'obbligo di sapere quanti saranno gli elementi nelle tue istruzioni. Spesso, però, o non lo sai o non vuoi disturbarti a contarli.
Talvolta la soluzione migliore per evitare un OUT OF DATA ERROR è di terminare la tua istruzione DATA con un INDICATORE. Un indicatore è un qualche valore che appare nel tuo elenco dei DATA ma che non serve ai fini della programmazione; questo può essere rappresentato da un numero negativo, un numero molto grande, un numero molto piccolo o una parola particolare come END o STOP. Quando usi un indicatore, devi ovviamente aggiungere un controllo, una istruzione IF/THEN, per dire al computer di diramarsi da un'altra parte del programma quando viene letto l'indicatore. Per esempio:

```
10 READ A
15 IF A < 0 THEN END
20 DATA 13, 35, 29, -999
25 PRINT "TOTALE = "; A
30 GOTO 10
```

Questo programma legge (READ) e stampa (PRINT) il valore di A fino a quando non arriva a -999. La riga 15 dice al computer di finire (END) immediatamente il programma quando viene letto un valore negativo. Ovviamente al posto dell'istruzione END si potrebbe mettere un numero di riga del programma (GOTO ##) in cui far andare il computer per proseguire con altro codice.

C'è anche un modo di riutilizzare gli elementi di un'istruzione DATA, riportando (RESTORE) il puntatore dei dati all'inizio dell'elenco DATA. Prova ad aggiungere questa riga:

```
45 RESTORE
```

al secondo programma di questo capitolo e rieseguilo (RUN) un'altra volta. Vedrai che il puntatore dei dati verrà ripristinato (RESTORE) sul primo elemento dell'elenco DATA, e potrai rileggere (READ) tutti gli elementi (in questo caso... all'infinito).

9.2. CALCOLO DI MEDIE

Qui c'è un programma che legge (READ) una serie di numeri da un elenco DATA e ne calcola la media. Questo programma usa anche un indicatore per dire al computer quando fermare la lettura dei dati.

```
 5 T = 0: CT = 0
10 READ X
20 IF X = -1 THEN 50: REM CONTROLLO PER L'INDICATORE
25 CT = CT + 1
30 T = T + X: REM AGGIORNA IL TOTALE
40 GOTO 10
50 PRINT "QUI C'ERANO"; CT ;"VALORI LETTI"
60 PRINT "TOTALE ="; T
70 PRINT "MEDIA ="; T/CT
80 DATA 75, 80, 62, 91, 87, 93, 78, -1

RUN
QUI C'ERANO 7 VALORI LETTI
TOTALE = 566
MEDIA = 80.8571429
```

La riga 5 imposta a zero CT, il ConTatore, e T, il Totale. La riga 10 legge (READ) un valore dall'elenco DATA e lo assegna a X.
La riga 20 controlla il valore letto su X per vedere se è il nostro indicatore (-1). Se sì, il programma scavalca le righe 25-40 e va dirittamente alla riga 50.
La riga 25 aggiunge una unità a CT, il contatore, solo se il valore di X non è l'indicatore.
La riga 30 somma X a T, il totale corrente. La riga 40 riporta indietro il programma per ripetere la riga 10. La riga 50, che non viene eseguita finché la riga 10 non legge (READ) l'indicatore, stampa (PRINT) la quantità di valori letti (CT).
La riga 60 stampa (PRINT) la somma dei numeri letti (T). La riga 70 stampa (PRINT) la media.
Nell'istruzione READ è possibile utilizzare anche più di una variabile. In questo caso sarà possibile mescolare i tipi di dati in un elenco DATA. Ecco un programma che fa, appunto, quanto appena detto. Esso legge (READ) un nome ed alcuni punteggi e poi calcola la media dei punteggi.

```
NEW

10 READ N$, A, B, C
20 PRINT "I PUNTEGGI DI ";N$;" SONO STATI :"; A ;" "; B ;" "; C
30 PRINT "E LA MEDIA È: "; (A + B + C) / 3
40 PRINT: GOTO 10
50 DATA MIKE, 190, 185, 165, DICK, 225, 245, 190
60 DATA JOHN, 155, 185, 205, PAUL, 160, 179, 187

RUN
I PUNTEGGI DI MIKE SONO STATI : 190   185   165
E LA MEDIA È : 180

I PUNTEGGI DI DICK SONO STATI : 225   245   190
E LA MEDIA È : 220
```

La riga 10 legge (READ) un valore per ognuna delle variabili. L'istruzione DATA elenca i suoi elementi nello stesso ordine in cui l'istruzione READ si aspetta di trovarli. In altre parole, c'è un nome che va con la variabile di stringa e numeri che vanno con le variabili intere.

9.3. VARIABILI INDICIZZATE

Fin qui abbiamo utilizzato solo semplici variabili del BASIC come X e X$. È fuori dubbio che non ti ritroverai a scrivere un programma che richieda più nomi di variabili di quante possano essere utte le combinazioni di lettere e numeri disponibili in BASIC, ma vorresti poter raggruppare insieme nomi di variabill da abbinare a gruppi di dati.
Le variabili indicizzate ti permettono di utilizzare nomi di variabili in modo da poterle facilmente raggruppare insieme. Per esempio:

A(0), A(1), A(2), A(3)

I numeri fra parentesi sono gli INDICI di A. Qui devi fare attenzione al fatto che la variabile chiamata A1 NON è uguale alla variabile indicizzata A(1). Come indice puoi utilizzare variabili ed operazioni aritmetiche. Per esempio:

A(X) A(X+1) A(4-1) A(2*X)

Le espressioni dentro le parentesi vengono valutate in base alle stesse regole riguardanti le operazioni aritmetiche descritte nel Capitolo 3.
Le variabili indicizzate, come le variabili semplici, nominano una locazione di memoria dentro il computer, ma solo i valori del nome delle variabili indicizzate che sono organizzati in una MATRICE.
Una MATRICE viene interprata dal computer come una unità, un elenco o una tabella, di valori relativi. L'esempio seguente utilizza le variabili indicizzate per calcolare una media:

```
 5 PRINT CHR$(147)
10 INPUT "QUANTI NUMERI :"; X
20 FOR A = 1 TO X
30 PRINT "IMMETTI IL VALORE #"; A ;: INPUT B(A)
40 NEXT
50 SU = 0
60 FOR A = 1 TO X
70 SU = SU + B(A)
80 NEXT
90 PRINT: PRINT "MEDIA ="; SU/X

RUN

QUANTI NUMERI :? 5
IMMETTI IL VALORE # 1 ? 125
IMMETTI IL VALORE # 2 ? 167
IMMETTI IL VALORE # 3 ? 189
IMMETTI IL VALORE # 4 ? 167
IMMETTI IL VALORE # 5 ? 158

MEDIA = 161.2
```

La riga 5 pulisce lo schermo.
La riga 10 ti chiede di inserire la quantità totale di elementi che immetterai (INPUT) alla riga 30.
La riga 20 configura un ciclo che rende A l'indice della matrice B. Il ciclo aggiunge una unità ad A ad ogni esecuzione e, in questo modo, aggiorna la matrice B.
La riga 30 ti chiede di immettere (INPUT) un valore per la varibile indicizzata B(A).
Il ciclo dalla riga 50 alla 80 sommano nella variabile SU tutti i numeri immessi con l'istruzione INPUT.
La riga 90 stampa (PRINT) la media.

Ogni volta che viene eseguito il ciclo INPUT, A viene incrementato di 1, per cui il valore successivo immesso verrà assegnato all'elemento successivo della matrice B. Alla fine del programma, la matrice B sarà simile a questa:

```
       +---------+
B(1)   |   125   |
       +---------+
B(2)   |   167   |
       +---------+
B(3)   |   189   |
       +---------+
B(4)   |   167   |
       +---------+
B(5)   |   158   |
       +---------+
```

Dopo aver immesso (INPUT) tutti i valori, questi verranno via via memorizzati nella matrice B. Adesso puoi accedere a questi valori utilizzando le variabili indicizzate. Per esempio, guarda cosa succede quando aggiungi queste righe:

```
100 PRINT B(X-1)
120 PRINT B(3)
130 PRINT B(X-3)
```

9.4. DIMENSIONAMENTO DELLE MATRICI

Se provi ad immettere più di dieci numeri in una matrice, riceverai un BAD SUBSCRIPT ERROR. Le matrici con più di dieci elementi devono essere predefinite con una istruzione di DIMensionamento. Per esempio, se hai bisogno di una matrice con 25 valori, dovrai scrivere questa istruzione nel tuo programma:

DIM B(25)

In una istruzione DIM puoi utilizzare anche una variabile. Per esempio: nell'ultimo programma è stata utilizzata questa istruzione poiché X corrispondeva alla quantità totale dei valori nella matrice B:

```
15 DIM B(X)
```

Però stai attento quando usi le variabili per definire le matrici: una volta dimensionata, una matrice non potrà essere ridimensionata da un'altra parte del programma. Per questo motivo, non utilizzare una variabile che nel corso del programma possa essere cambiato il suo valore. In un programma puoi inserire più di una matrice e le potrai DIMensionare tutte sulla stessa riga:

```
10 DIM A(12), B(35), C(3,5)
```

Le matrici A e B sono mono-dimensionali, ma C è una matrice bi-dimensionale. Le matrici mono-dimensionali hanno solo righe di dati (in pratica come una normalissima variabile), ma le matrici bi-dimensionali hanno sia righe che colonne di dati, proprio come una tabella. La matrice C ha 3 righe e 5 colonne. In una istruzione di DIMensionamento le righe vengono sempre elencate per prime.

9.5. ROTOLAMENTO SIMULATO DI DADO CON LE MATRICI

Via via che inizi a scrivere programmi più complessi, troverai che le variabili indicizzate riducono la quantità di istruzioni e rendono i programmi più semplici da scrivere.
Per esempio: una sola variabile indicizzata può tenere traccia della quantità di volte che esce una faccia di un dado fatto rotolare:

```
 1 PRINT CHR$(147) : REM ROTOLAMENTO DADO
10 INPUT "QUANTI ROTOLAMENTI: "; X
20 FOR L = 1 TO X
30 R = INT(6*RND(1)) + 1
40 F(R) = F(R) + 1
50 NEXT L
60 PRINT "FACCIA", "QUANTITA' DI VOLTE"
70 FOR C = 1 TO 6: PRINT C, F(C): NEXT
```

La riga 10 ti chiede quante volte lancerai il dado nel rotolamento simulato. La riga 20 configura un ciclo per contare la quantità di rotolamenti del dado in modo da terminare il programma sul Xmo rotolamento.
La riga 30 inserisce in R il numero casuale che è risultato dal rotolamento.
La riga 40 configura la matrice F (di Faccia), che tiene traccia di quante volte è uscita ciascuna faccia del dado. Qualsiasi valore acquisito da R nel rotolamento del dado, diventa l'indice della matrice, e la riga 40 aggiunge un uno alla variabile adatta della matrice. Per esempio: ogni volta che esce un 2, F(2) viene incrementato di uno.
La riga 70 stampa (PRINT) la quantità di volte che è uscita ogni faccia. Ecco un'esecuzione del programma:

```
QUANTI ROTOLAMENTI: ? 1000
FACCIA    QUANTITA' DI VOLTE
 1          148
 2          176
 3          178
 4          166
 5          163
 6          169
```

Adesso ti mostreremo quanto sarebbe più lungo il tuo programma senza utilizzare una matrice:

```
10 INPUT "QUANTI ROTOLAMENTI: "; X
20 FOR L = 1 TO X
30 R = INT(6*RND(1)) + 1
40 IF R = 1 THEN F1 = F1 + 1: NEXT
41 IF R = 2 THEN F2 = F2 + 1: NEXT
42 IF R = 3 THEN F3 = F3 + 1: NEXT
43 IF R = 4 THEN F4 = F4 + 1: NEXT
44 IF R = 5 THEN F5 = F5 + 1: NEXT
45 IF R = 6 THEN F6 = F6 + 1: NEXT
60 PRINT "FACCIA", "QUANTITA' DI VOLTE"
70 PRINT 1, F1
71 PRINT 2, F2
72 PRINT 3, F3
73 PRINT 4, F4
74 PRINT 5, F5
75 PRINT 6, F6
```

Come puoi vedere, il programma ha il doppio di righe. Quanto usi le matrici puoi risparmiare in lunghezza, spazio e tempo nel fare il programma.

9.6. MATRICI BI-DIMENSIONALI

Come abbiamo accennato in precedenza, le matrici bi-dimensionali hanno sia righe che colonne, come un diagramma o una tabella. Ovviamente le matrici bi-dimensionali hanno due indici: il primo serve per il numero di RIGA, il secondo serve per il numero di COLONNA. Per esempio:

A(4,6) ha 4 RIGHE e 6 COLONNE

Ecco come potrebbe apparire in memoria la matrice A se paragonata ad una griglia bi-dimensionale:

```
          0       1       2       3       4       5       6
      +-------+-------+-------+-------+-------+-------+-------+
  0   |       |       |       |       |       |       |       |
      +-------+-------+-------+-------+-------+-------+-------+
  1   |       |       |       |       |       |       |       |
      +-------+-------+-------+-------+-------+-------+-------+
  2   |       |       |       |       |       |       |       |
      +-------+-------+-------+-------+-------+-------+-------+
  3   |       |       |       |       |       |       |       |
      +-------+-------+-------+-------+-------+-------+-------+
  4   |       |       |       |       |       |       |       |
      +-------+-------+-------+-------+-------+-------+-------+
```

Noterai che c'è una riga ed una colonna con indice 0; quando dimensioni A(4,6), in pratica stai creando una matrice con 5 righe e 7 colonne, oppure 35 elementi.
In una matrice bi-dimensionale puoi accedere a qualsiasi elemento utilizzando i suoi indici di riga e colonna. Per esempio: supponiamo di voler assegnare 255 ad A(3,4):

```
10 LET A(3,4) = 255
```

Ecco come apparirà adesso la nostra griglia:

```
          0       1       2       3       4       5       6
      +-------+-------+-------+-------+-------+-------+-------+
  0   |       |       |       |       |       |       |       |
      +-------+-------+-------+-------+-------+-------+-------+
  1   |       |       |       |       |       |       |       |
      +-------+-------+-------+-------+-------+-------+-------+
  2   |       |       |       |       |       |       |       |
      +-------+-------+-------+-------+-------+-------+-------+
  3   |       |       |       |       |  255  |       |       |
      +-------+-------+-------+-------+-------+-------+-------+
  4   |       |       |       |       |       |       |       |
      +-------+-------+-------+-------+-------+-------+-------+
```

Le matrici bi-dimensionali seguono le stesse regole delle matrici ad una dimensione:

DIMensionamento:	DIM A(20,20)
Assegnazione di valori:	A(1,1) = 255
Assegnazione dei valori ad altre variabili:	AB = A(1,1)
Stampa dei valori:	PRINT A(1,1)

Ecco qui un esempio di come è possibile utilizzare le matrici bi-dimensionali. Questo esempio tiene traccia delle risposte a un questionario di club come questo:

QUESTIONARIO DI CLUB

Q1: SEI A FAVORE DELLA DECISIONE #1?

1-SI   2-NO   3-NON SO

Supponiamo che ci siano quattro domande, in modo da dimensionare una matrice, che chiameremo A, come A(4,3). Ecco a cosa assomiglia la tabella della matrice:

```
                          SI                 NO                NON-SO

                    +----------------+----------------+----------------+
   DOMANDA 1        |                |                |                |
                    +----------------+----------------+----------------+
   DOMANDA 2        |                |                |                |
                    +----------------+----------------+----------------+
   DOMANDA 3        |                |                |                |
                    +----------------+----------------+----------------+
   DOMANDA 4        |                |                |                |
                    +----------------+----------------+----------------+
```

Il programma che tiene traccia delle risposte è qui sotto. Questo programma utilizza molte delle tecniche di programmazione che sono state presentate fin qui.
Le righe dalla 30 alla 65 stampano (PRINT) le domande in ordine numerico e ti chiedono di immettere (INPUT) la risposta.
La riga 70 aggiunge uno all'elemento adatto della matrice. Ricorda che R è il numero della domanda, e le domande si trovano nelle righe. C è il numero della risposta, e le risposte si trovano nelle colonne.
La riga 90 chiede se hai un'altra serie di risposte da immettere (INPUT).
Le righe 110 e 120 dicono al programma dove andare in base alla tua risposta sulla riga 90.
Le righe da 130 a 170 stampano (PRINT) la quantità totale di ciascuna risposta.
Ogni volta che immetti (INPUT) una risposta sulla riga 61, la riga 70 aggiorna il giusto elemento della matrice. Richiamare la R è il numero della domanda e C è il numero della risposta, così se la tua risposta alla domanda 2 fosse 3 (non so), la riga 70 aggiunge uno all'elemento A(2,3) della matrice.
Noterai che in questo esempio non abbiamo utilizzato la riga e la colonna con indice 0. Non devi necessariamente utilizzare questa riga e colonna, ma ricorda che in ogni matrice che utilizzi saranno sempre presenti.

```
 20 PRINT "{CLR/HOME}"
 30 FOR R = 1 TO 4
 40 PRINT "DOMANDA # :"; R
 50 PRINT "1-SI  2-NO  3-NON SO"
 60 PRINT "QUAL'E' STATA LA RISPOSTA : ";
 61 GET C: IF C < 1 OR C > 3 THEN 61
 65 PRINT C: PRINT
 70 A(R,C) = A(R,C) + 1: REM AGGIORNA ELEMENTO
 80 NEXT R
 85 PRINT
 90 PRINT "VUOI INSERIRE UN'ALTRA": PRINT "RISPOSTA (S/N) ?";
100 GET A$ : IF A$ = "" THEN 100
110 IF A$ = "S" THEN 20
120 IF A$ <> "N" THEN 100
130 PRINT "{CLR/HOME}"; "LE RISPOSTE TOTALI SONO STATE:": PRINT
140 PRINT SPC(18); "RISPOSTA"
141 PRINT "DOMANDA", "SI", "NO", "NON SO"
142 PRINT "--------   -----------------------------"
150 FOR R = 1 TO 4
160 PRINT R, A(R,1), A(R,2), A(R,3)
170 NEXT R
```

```
RUN

DOMANDA # : 1
1-SI   2-NO   3-NON SO
QUAL'E' STATA LA RISPOSTA : 1

DOMANDA # : 2
1-SI   2-NO   3-NON SO
QUAL'E' STATA LA RISPOSTA : 1

E così via...

VUOI INSERIRE UN'ALTRA
RISPOSTA (S/N) ?

LE RISPOSTE TOTALI SONO STATE:

                 RISPOSTE
DOMANDA      SI        NO         NON SO
--------    -----------------------------
 1           6          1          0
 2           5          2          0
 3           7          0          0
 4           2          4          1
```

APPENDICI

INTRODUZIONE

Adesso che sei diventato più intimamente coinvolto con il tuo Commodore 64, noi vogliamo farti sapere che il nostro supporto al cliente non finisce qui. Tu non lo sai, ma la Commodore è stata in affari per oltre 23 anni. Negli anni 70 abbiamo introdotto il primo computer personale, fine a se stesso, (il PET). Da allora siamo diventati la società di computer dominante in molti paesi del mondo. La nostra abilità nel progettare e produrre i nostri circuiti integrati per computer ci permettono di portarti computer personali sempre più nuovi e migliori a prezzi che mai ti aspetteresti per questo livello di eccellenza tecnica. La Commodore è impegnata nel sostegno non solo di te, l'utente finale, ma anche il rivenditore dal quale hai acquistato il tuo computer, i periodici che pubblicano articoli 'come-fare' che mostrano le nuove applicazioni o tecniche, e... ancor più importante... gli sviluppatori di software che presentano programmi su cartuccia, disco e nastro da utilizzare sul tuo computer. Da parte nostra ti sollecitiamo ad istituirti o a prender parte ad un "circolo utente" Commodore in cui poter imparare nuove tecniche, scambiare idee e condividere le scoperte. Noi pubblichiamo due periodici indipendenti che contengono suggerimenti sulla programmazione, informazioni sui nuovi prodotti ed idee per le applicazioni sul computer. (Vedi l'Appendice N).
In America del nord la Commodore fornisce una "Rete di Informazioni Commodore" sul CompuServe Information Service... per accedere a questa rete, tutto ciò che ti serve è il tuo computer Commodore 64 ed la nostra cartuccia a basso costo VICMODEM per l'interfaccia telefonica (oppure un altro modem compatibile).

Le APPENDICI che seguono comprendono diagrammi, tabelle ed altre informazioni che ti aiuteranno a programmare il tuo Commodore 64 in modo più veloce e più efficiente. Inoltre comprendono importanti informazioni sul vasto assortimento di prodotti Commodore, ai quali potresti essere interessato, ed un elenco bibliografico di oltre 20 libri e periodici che ti possono aiutare a sviluppare le tue capacità nella programmazione e tenerti al corrente sulle informazioni più recenti riguardanti il tuo computer e le unità periferiche.

APPENDICE A

ESPANSIONE DEL TUO COMPUTER COMMODORE 64

Il 64 è un computer estremamente potente e che può essere utilizzato in un'ampia varietà di applicazioni, dalla elaborazione di tesi alla gestione di archivi di dati.
Il sistema principale del 64 consiste nel computer, un televisione adeguato o monitor ed una unità a cassetta sulla quale memorizzare i tuoi programmi. Comunque, per alcune applicazioni l'unità a cassetta potrebbe rivelarsi piuttosto lenta. Questa limitazione può essere superata utilizzando una unità a disco in cui salvare e richiamare i tuoi programmi.

Perché Usare un Gestore Disco?

Se i tuoi programmi o file di dati sono molto piccoli, la memorizzazione delle informazioni su nastro a cassetta non causerà nessun disagio. Tuttavia, arriverà un momento in cui l'impiego efficace del tuo sistema verrà ostacolato dal tempo occorrente per caricare da e/o salvare su nastro. Questo è il momento in cui dovresti pensare di utilizzare una unità a disco della Commodore.

IL GESTORE DISCO VIC 1541

Il gestore disco VIC 1541 ti permette di memorizzare fino a 144 programmi o file di dati su un disco standard da 5,25 pollici. Quando utilizzi un gestore disco, non dovrai più pensare a riavvolgere i nastri o preoccuparti di sovrascrivere informazioni esistenti. L'unità a disco 1541 è un 'dispositivo intelligente', nel senso che esegue tutte le sue operazioni senza utilizzare nessuna risorsa della memoria del COMMODORE 64. Su ciascun dischetto è possibile memorizzare oltre 174000 caratteri di informazione -- la dimensione di un romanzo medio di Dickens! Il vantaggio principale di un gestore disco su una unità a cassetta è comunque la velocità. Di solito un'operazione su un gestore disco è fino a 40 volte più veloce della stessa operazione su cassetta. L'unità disco VIC 1541 non richiede alcuna interfaccia particolare perché si collega direttamente al tuo COMMODORE 64. Non sei limitato nemmeno a utilizzare un solo gestore disco. Con un solo COMMODORE 64 è possibile collegare insieme (chiamato 'concatenamento a margherita') fino a cinque gestori disco; in modo da poter caricare e salvare programmi e file senza dover sostituire i dischetti.

STAMPANTE A MATRICE DI PUNTI VIC 1525

Una stampante aggiunge moltissima versatilità al tuo sistema computerizzato. Nessun computer è completo senza quella. La stampante ti permette di produrre fatture, spedire lettere o stampare listati di programma in modo da poter esaminare il tuo codice al di fuori dal computer. La VIC 1525 è una stampante a 'matrice di punti', il che vuol dire che ogni carattere viene formato da un modello di punti in una griglia. Puoi stampare tutti i caratteri sulla tastiera del tuo COMMODORE 64 oppure altri caratteri disegnati da te. La VIC 1525 stampa ad una velocità di 30 caratteri/secondo con 12 caratteri/pollice su carta comune con avanzamento a trattore, larga fino a 10 pollici. La stampante si collega direttamente al tuo COMMODORE 64 e non richiede nessuna ulteriore interfaccia.

STAMPANTE BI-DIREZIONALE A MATRICE DI PUNTI VIC 1526

La stampante VIC 1526 differisce dalla VIC 1525 in due cose importanti. Prima di tutto la 1526 è una stampante bi-direzionale, il che vuol dire che la macchina stampa da destra a sinistra allo stesso modo in cui stampa tradizionalmente da sinistra a destra. Facendo così la macchina non spreca tempo a mandare la testina di stampa sul margine sinistro del foglio ogni volta che viene stampato un 'a capo' del testo.
L'altra differenza principale tra la 1526 e la 1525 è nella velocità delle operazioni. Il tempo necessario a stampare ciascuna riga è direttamente proporzionale alla larghezza di pagina che hai configurato. Con una stampa di 80 colonne la velocità è di 45 righe/minuto; su 40 colonne 78 righe/minuto e, su un foglio largo 20 colonne, 124 righe/minuto. La 1526 ti permette di stampare fino a 3 copie della tua uscita, compreso l'originale. Ha un nastro che si può cambiare facilmente ed accetta fogli larghi fino a 10 pollici. La 1526 si collega direttamente al tuo COMMODORE 64 senza alcuna interfaccia speciale.

STAMPANTE/PLOTTER VIC 1520

Il VIC 1520 può essere utilizzato sia come stampante standard che come tracciatore grafico per permetterti di disegnare e tracciare grafici, istogrammi, diagrammi a torta -- in realtà qualsiasi forma che tu voglia, in una combinazione di quattro colori. Puoi stampare lettere maiuscole/minuscole e simboli grafici. Una ulteriore agevolazione offerta dalla stampante ti permette di definire il corpo (la dimensione) di ogni carattere da visualizzare. La stampante stampa 14 caratteri/secondo. In base al corpo scelto per il carattere, tra 10 e 40, i caratteri possono essere stampati su ogni riga del foglio. I caratteri possono essere stampati anche 'obliquamente' ruotandoli di 90 gradi. Le forme vengono tracciate semplicemente dando alla stampante/plotter le coordinate iniziali e finali o la forma che deve andare sul foglio e quale tipo di linea vuoi utilizzare. In questo caso la linea varia da solida a comune linea tratteggiata. Il tracciatore grafico è accurato fino a 0,2 mm ed ha una velocità di tracciamento di 60 mm/secondo. Il tracciatore grafico utilizza piccole penne a sfera che sono selezionabili dall'utente durante il tracciamento. La stampante/plotter si collega direttamente al tuo COMMODORE 64 senza alcuna ulteriore interfaccia.

MONITOR A COLORI 1701

Un computer con la versatilità del COMMODORE 64 ha bisogno di un mezzo sul quale dimostrare al massimo le sue capacità. Il monitor a colori 1701 è stato progettato in particolare per questo scopo. Il monitor ha uno schermo di 14 pollici con una risoluzione notevole. Il suono può essere prodotto o dagli altoparlanti interni del monitor oppure, attraverso un semplice collegamento, dal tuo sistema Hi-Fi.

JOYSTICK 1311

Un joystick può essere utilizzato non solo come controllore di giochi ma anche, con un software adeguato, come strumento per il disegno ed il tracciamento. La Guida di Riferimento al Programmatore del COMMODORE 64 dà informazioni dettagliate su come unire l'impiego di un joystick nei tuoi programmi.

SOFTWARE

Per il COMMODORE 64 è disponibile un'ampia varietà di software che coprono applicazioni per casa, per lavoro, intrattenimento ed assistenze per il programmatore.

Software per Affari e per Casa

EASYFILE EFI 6440 (disco)

EASYFILE è un sistema comprensibile di data-base, per il COMMODORE 64, che comprende tutte le caratteristiche di altri pacchetti di costo elevato, ma ad una frazione di prezzo. L'utente decide come vuole far apparire le sue informazioni quando vengono stampate o a schermo o sulla stampante. Questo significa che EASYFILE può essere personalizzato per soddisfare le necessità di un'ampia varietà di applicazioni, sia in affari che per casa.

CLUB MANAGER CMG 6440 (disco)

CLUB MANAGER, con il suo funzionamento agevole, è stato concepito per aiutare circoli sportivi, circoli sociali, associazioni -- in pratica tutte quelle organizzazioni che hanno bisogno di mantenere accurati record delle appartenenze. Il pacchetto permette di registrare su disco i particolari di tutti i soci del club dato che i record vengono memorizzati all'interno di una cartellina di archiviazione. Sarà possibile rettificare i particolari di appartenenza, dove necessario, mentre i record potranno essere aggiunti o cancellati dal file di appartenenza. CLUB MANAGER permette di presentare elenchi di

appartenenza, promemorie di sottoscrizione, etichette d'indirizzi e, affinché il pacchetto possa essere collegato all'elaboratore di testi EASY SCRIPT, copie personalizzate di lettere standard.
CLUB MANAGER possiede anche la facoltà di prenotazioni permettendo di inserire i dettagli di quando i soci vogliono utilizzare i campi da squash, i campi da tennis, le tavole di snooker, le tavole per il ristorante, i biglietti per il ballo o qualsiasi altra attività similare da club. Adesso la scrittura sui libri è una cosa del passato. CLUB MANAGER permette di mantenere un'agenda e la utilizza per registrare/correggere i particolari di appuntamenti, incontri, ecc. CLUB MANAGER è lo strumento ideale per i proprietari o i segretari di club e riduce enormemente la quantità di tempo passato per il mantenimento degli archivi degli appartenenti e per la facilità delle prenotazione del club.

FUTURE FINANCE FFI 6440 (disco)

FUTURE FINANCE è un pacchetto a basso costo per progettazione finanziaria. Esso permette di prevedere i profitti di una società e la posizione del flusso di cassa in base alla produzione prevista, alle vendite ed ai costi. È possibile modificare i dettagli per vedere l'effetto di queste variazioni sul rendimento generale della società. È possibile presentare i rapporti che mostrano l'apporto ai profitti di ogni prodotto e la posizione del flusso di cassa alla fine di ogni periodo definito dall'utente.

EASYCALC ECL 6440 (disco)

EASYCALC è un pacchetto che contiene un foglio di calcolo elettronico. Esso comprende tutte le caratteristiche dei fogli di calcolo tradizionali -- grandezza del foglio definibile dall'utente, copia delle informazioni da un'area del foglio all'altra, ecc. Insieme a queste agevolazioni, EASYCALC comprende molte altre caratteristiche incluse in una libreria di trigonometrici, funzioni statistiche ed altre funzioni matematiche progredite, la possibilità di tracciare grafici in un'area ben precisa del foglio, protezione dei dati attraverso una opzione a parola d'ordine, e molto, molto altro ancora.

EASY STOCK EST 6440 (disco)

EASY STOCK è un potente sistema di inventario che contiene un'ampia serie di registrazione dei rifornimenti e riportandone le caratteristiche. I particolari di ciascun elemento di rifornimento viene immesso direttamente da tastiera sullo schermo e poi memorizzato su dischetto. EASY STOCK permette anche di modificare il prezzo di un singolo elemento o una serie di prodotti sul file dei rifornimenti. È possibile accedere facilmente a ciascuno record dei rifornimenti immettendo il numero di riferimento/pezzo di quell'elemento. I record possono essere corretti, aggiunti al tuo file dei rifornimenti oppure cancellati. Se la quantità di giacenza si abbassa sotto il livello minimo indicato, il rifornimento rappresentato viene evidenziato ogni volta che si accede a record. EASY STOCK permette di presentare un vasto campo di rapporti che comprendono livelli di rifornimento, analisi dei movimenti di rifornimento, analisi valutativa sulle vendite/rifornimenti ed altro ancora.

EASY SCRIPT ESC 6440 (disco)

EASY SCRIPT è un pacchetto professionale, di basso costo, per l'elaborazione del testo. Esso permette di creare, modificare e stampare rapidamente e facilmente il testo. EASY SCRIPT può essere utilizzato per scrivere lettere, rapporti, promemorie, manoscritti di libro -- in pratica qualsiasi tipo di documento. Il testo può essere memorizzato su dischetto o cassetta in modo da poterlo stampare o modificare in un secondo tempo.

EASY SPELL ESP 6440 (disco)

EASY SPELL è un controllore ortografico per file prodotti dal pacchetto di elaborazione testi EASY SCRIPT. Esso può essere utilizzato per esaminare il testo in singoli file di EASY SCRIPT oppure il testo distribuito in file collegati insieme. La confezione di EASY SPELL è completa di un dischetto-dizionario con il quale viene controllata l’ortografia del testo.

Aiuti per il Programmatore

Per assistere nello sviluppo del tuo software personale, la Commodore ha introdotto una serie di utilità per la programmazione. Questi programmi ti aiuteranno a velocizzare l'immissione e la depurazione del BASIC e dei programmi in codice macchina.

SIMON'S BASIC SIB 6410 (cartuccia)

SIMON'S BASIC è stato concepito per permettere ai programmatori, di tutti i livelli, di utilizzare facilmente la potenzialità del loro COMMODORE 64. In realtà la cartuccia del SIMON'S BASIC è tre pacchetti in uno. Essa contiene un Toolkit per rimuovere i noiosi aspetti di programmazione del computer, un'ampia serie di comandi per facilitare l'impiego di grafiche e suoni sul 64, e comandi per la Programmazione Strutturata per aiutare il programmatore a scrivere codice più significativo. Il pacchetto è fornito nella forma a cartuccia, il che significa che puoi utilizzare tutte le sue funzioni semplicemente inserendola nella fessura posteriore del COMMODORE 64 ed accendendo il computer -- è proprio semplice quanto quello appena detto. In questo caso sarà possibile utilizzare i comandi supplementari del SIMON'S BASIC esattamente come qualsiasi altro comando del BASIC.

I comandi del Toolkit comprendono:

- ✓ AUTO - per la numerazione automatica delle righe di programma
- ✓ RENUMBER - per una rinumerazione automatica del programma
- ✓ KEY - per assegnare i comandi ai tasti funzione
- ✓ e molto altro ancora.

I comandi per la grafica comprendono:

- ✓ HIRES - per mettere lo schermo in modalità alta risoluzione
- ✓ REC - per tracciare una forma rettangolare
- ✓ CIRCLE - per tracciare una forma circolare
- ✓ PAINT - per riempire di colore una sagoma

Più i comandi per la creazione di sprite e la grafica definita dall'utente:

- ✓ DESIGN - per configurare una griglia di disegno per uno sprite o carattere definibile dall'utente
- ✓ MMOB - per muovere uno sprite
- ✓ DETECT - per rilevare la collisione fra sprite

…e molto, molto altro ancora.

I comandi per la Programmazione Strutturata, forniti dalla cartuccia del SIMON'S BASIC, sono un vantaggio per i programmatori di tutti livelli di abilità. Adesso è possibile etichettare le routine BASIC e richiamare queste routine per nome. Altri comandi per la programmazione strutturata comprendono:

- ✓ PROC - per etichettare le routine BASIC
- ✓ CALL - per passare l'esecuzione ad una routine
- ✓ EXEC - per passare l'esecuzione ad una routine e ritornare da essa non appena la routine ha terminato il suo compito
- ✓ REPEAT..UNTIL - per ripetere una sequenza in base alla verifica di una condizione
- ✓ e molti altri.

SIMON'S BASIC comprende anche comandi per formattazione dello schermo, lo scorrimento dello schermo, la convalida d'immissione, la gestione di stringhe di caratteri, la conversione da esadecimale a decimale e da binario a decimale, la divisione di interi e molto, molto ancora. La cartuccia inoltre possiede un gruppo di comandi che permettono di intrappolare certi errori BASIC. È possibile anche creare i propri messaggi d'errore! La serie di comandi, fornita dalla cartuccia del SIMON'S BASIC, lo rende uno strumento essenziale per ogni programmatore che voglia utilizzare facilmente le funzioni speciali del suo COMMODORE 64.
Lavora con cassetta o dischetto.

ASSEMBLER TUTOR AST 6440 (disco) AST 6420 (cassetta)

Il pacchetto ASSEMBLER TUTOR è una necessità per tutti quelli che vogliono programmare in codice macchina, ma può essere prezioso anche per quei programmatori che sanno già qualcosa di programmazione del linguaggio assembly e vogliono espandere la loro conoscenza sul codice macchina del 6502. L'ASSEMBLER TUTOR è suddiviso in tre moduli, ciascuno dei quali ricopre un aspetto sulla programmazione in linguaggio assembly e comprende una introduzione, un auto-analisi ed una discussione sui vari aspetti della programmazione in linguaggio assembly.

ASSEMBLER DEVELOPMENT ASM 6440 (disco)

Il pacchetto ASSEMBLER DEVELOPMENT permette di programmare in assembler direttamente sul COMMODORE 64. Esso fornisce tutti gli strumenti necessari al programmatore assembler per creare, caricare l'assembler ed eseguire il codice del linguaggio assembly del 6510.

PROGRAMMER'S UTILITIES UTL 6440 (disco)

Il pacchetto PROGRAMMER'S UTILITIES comprende molte routine utili per aiutare il programmatore nuovo ed esperto a dare il massimo sul COMMODORE 64. Queste utilità comprendono le routine Disk-Handling per cambiare il numero di dispositivo di un gestore disco e duplicare dischi usando un solo gestore disco, utilità grafiche come un editore di sprite e di caratteri, aiuti sui comandi per il suono e la programmazione BASIC per una formattazione dello schermo più facile ed operazioni di controllo.

INTRATTENIMENTO

Allo stesso modo di una vasta gamma di giochi di qualità alta tipo arcade per il COMMODORE 64, la Commodore ha prodotto anche una serie di programmi di Simulazione per quelli che preferiscono esercitare il cervello invece del dito sul pulsante di fuoco del joystick. In questi giochi il tempo è illimitato, e si esce da scenari complessi solo con le proprie capacità.

LABYRINTH LBY 6420 (cassetta)

Osi entrare nel contorto labirinto Elizabettiano? Ne uscirai mai, o rimarrai intrappolato per sempre? Ti sei PERSO? Beh, puoi dare una rapida sbirciatina per vedere dove ti trovi ma, ricorda che, così facendo, si riduce il tuo punteggio. Certamente un gioco in cui perdersi.

HIGH FLYER HFL 6440 (disco)

È il 1945. La Guerra è appena terminata ed hai deciso di avviare una tua compagnia aerea. Facendo accurate decisioni gestionali, devi guidare l'impresa dal 1945 fino ai giorni nostri. Le decisioni sono tutte tua -- itinerari e programmi aerei; quale sia il miglior rapporto carico/passeggeri per massimizzare i profitti; quando sia il momento migliore per amplare la tua flotta, migliorare la scorta esistente, migliorare i servizi di supporto; se devi o no prendere denaro dalle banche e quale sia la banca che ti potrà offrire l'accordo migliore, ecc. Puoi salvare la posizione che hai raggiunto fino a quando, la volta successiva, vorrai unire gli aviatori migliori. Disponibile solo su disco.

RAIL BOSS RBO 6440 (disco)

Sei un pioniere ferroviario con l'incarico di costruire una linea nell'ovest americano tra Base City e Junction City. Il tuo compito è di assumere e licenziare geometri, lavoratori e guardie per proteggere te ed i tuoi operai contro banditi predoni. Costruendo stazioni lungo il percorso, puoi produrre reddito per pagare i tuoi operai ed acquistare ulteriori rifornimenti richiesti da tutta l'impresa. Per contrastare i tuoi lavori, indiani bellicosi cercano di bloccare il tuo lavoro uccidendo operai e strappando i binari

dalla sede. La tua sola speranza è che la cavalleria del Fort Commodore possa arrivare agli indiani prima che arrivino a te. Un gioco per persone che si sentono pionieri.

OCEAN RACER OCR 6440 (disco)

Sei entrato nella gara per il giro del mondo in yacht. La gara inizia da Portsmouth e passa attraverso Cape Town in Sud Africa, Auckland in Nuova Zelandia, Rio de Janeiro in Brasile e, infine, ritorna a Portsmouth. Puoi scegliere il tipo di barca che vuoi capitanare: un cutter/sloop ad albero singolo; un ketch a doppio albero oppure una imbarcazione a più scafi a singolo albero. La gara si svolge in quattro fasi, ognuna delle quali comprende vari rischi che variano dagli iceberg a danni all'imbarcazione causati dal passaggio di balene! Tu decidi quale rotta prendere e quali/quante vele scegliere in base alle condizioni del vento. Un gioco per vecchi lupi di mare ed aspiranti marinai.

Questi sono solo i primi di una serie di giochi di abilità particolarmente designati per il COMMODORE 64. Ma ce ne sono sempre di più in sviluppo. Per conoscere le date di distribuzione si prega di stare sintonizzato con il commerciante locale della Commodore.

APPENDICE B

DESCRIZIONE DEI MESSAGGI D'ERRORE DEL DOS

NOTA: I numeri dei messaggi d'errore inferiori al 20 dovrebbero essere ignorati con l'eccezione dello 01 che dà informazioni sulla quantità di file irregolari con il comando SCRATCH.

20: READ ERROR (block header not found)
Il controllore del disco è impossibilitato ad individuare l'intestazione del blocco-dati richiesto. Causato da un numero illegale di settore, o dall'intestazione che è stata rovinata.

21: READ ERROR (no sync character)
Il controllore del disco è impossibilitato a rilevare una marcatura di sincronizzazione sulla traccia richiesta. Causato dal disallineamento della testina di lettura/scrittura, nessuna presenza di dischetto, o disco non formattato o inserito impropriamente. Indica anche un guasto hardware.

22: READ ERROR (data block not present)
Al controllore del disco è stato richiesto di leggere o verificare un gruppo di dati che non è stato scritto correttamente. Questo messaggio d'errore capita insieme ai comandi BLOCK ed indica una traccia illegale e/o richiesta di gruppo.

23: READ ERROR (checksum error in data block)
Questo messaggio d'errore indica che c'è un errore in uno o più byte nei dati. I dati sono stati letti nella memoria del DOS, ma la somma di controllo sui dati è in errore. Questo messaggio può indicare anche problemi di fondo.

24: READ ERROR (byte decoding error)
Il dato o l'intestazione sono stati letti nella memoria del DOS, ma è avvenuto un errore hardware a causa di un modello di bit non corretto nel byte dei dati. Questo messaggio può indicare anche un errore di fondo.

25: WRITE ERROR (write-verify error)
Questo messaggio viene prodotto se il controllore rileva un disadattamento tra i dati scritti ed i dati nella memoria del DOS.

26: WRITE PROTECT ON
Questo messaggio viene prodotto quando al controllore è stato richiesto di scrivere un gruppo di dati con inserita la protezione in scrittura. Normalmente questo è causato utilizzando un dischetto con la tacca di protezione da scrittura coperta con un'etichetta.

27: READ ERROR (checksum error in header)
Il controllore ha rilevato un errore nell'intestazione del gruppo di dati richiesto. Il gruppo non è stato letto nella memoria del DOS. Questo messaggio può indicare anche problemi di fondo.

28: WRITE ERROR (long data block)
Il controllore cerca di rilevare il segno di sincronizzazione dell'intestazione successiva dopo aver scritto un gruppo di dati. Se detto segno non appare entro un tempo stabilito, viene prodotto il messaggio d'errore. L'errore è causato da un formato difettoso del dischetto (i dati si estendono nel blocco successivo), o da un guasto hardware.

29: DISK ID MISMATCH
Questo messaggio viene prodotto quando al controllore viene richiesto di accedere ad un dischetto che non è stato inizializzato. Il messaggio può capitare anche se un dischetto ha una intestazione difettosa.

30: SYNTAX ERROR (general syntax)
Il DOS non riesce ad interpretare il comando inviato al canale dei comandi. Normalmente questo è provocato da una quantità illegale di nomi di file o modelli illegalmente usati. Per esempio, sul lato sinistro del comando COPY possono apparire due nomi di file.

31: SYNTAX ERROR (invalid command)
Il DOS non riconosce il comando. Il comando deve cominciare dalla prima posizione.

32: SYNTAX ERROR (invalid command)
Il comando inviato è più lungo di 58 caratteri.

33: SYNTAX ERROR (invalid file name)
Nei comandi OPEN e SAVE viene usata scorrettamente l'uguaglianza del modello.

34: SYNTAX ERROR (no file given)
Da un comando è stato omesso il nome di un file o il DOS non lo riconosce come tale. Normalmente fuori dal comando sono stati dimenticati i due punti (:) di separazione.

39: SYNTAX ERROR (invalid command)
Questo errore può risultare se il comando inviato al canale dei comandi (indirizzo secondario 15) non viene riconosciuto dal DOS.

50: RECORD NOT PRESENT
Effetto di lettura del disco oltre l'ultimo record attraverso i comandi INPUT# o GET#. Questo messaggio capita anche dopo il posizionamento di un record oltre la fine del file in un file relativo. Se l'intenzione è di ampliare il file aggiungendo il nuovo record (con un comando PRINT#), il messaggio d'errore può essere ignorato. Dopo questo errore INPUT o GET non dovranno essere tentati perché verranno rilevati senza riposizionarli.

51: OVERFLOW IN RECORD
L'istruzione PRINT# eccede i limiti del record. L'informazione verrà troncata. Poiché il ritorno di carrello (Invio), che viene dato come un termine del record, viene incluso nella dimensione del record, uscirà questo messaggio se i caratteri totali nel record (compreso l'ultimo ritorno di carrello) eccederanno la dimensione definita.

52: FILE TOO LARGE
La posizione del record dentro un file relativo indica che ne risulterà la sovradimensione del disco.

60: WRITE FILE OPEN
Questo messaggio viene prodotto quando un file in scrittura, che non è stato chiuso, deve essere aperto in lettura.

61: FILE NOT OPEN
Questo messaggio viene prodotto quando si deve accedere ad un file che non è stato aperto nel DOS. Qualche volta, in questo caso, non viene prodotto un messaggio; la richiesta viene normalmente ignorata.

62: FILE NOT FOUND
Il file richiesto non esiste sul dispositivo indicato.

63: FILE EXISTS
Sul disco esiste già il nome del file che sta per essere creato.

64: FILE TYPE MISMATCH
Il tipo di file non è uguale al tipo di file nella cartella per il file richiesto.

65: NO BLOCK
Questo messaggio avviene con il comando B-A. Esso indica che il blocco da assegnare è stato già assegnato in precedenza. I parametri indicano la traccia ed il settore disponibili con il numero successivo più alto. Se i parametri sono a zero (0), saranno in uso tutti i blocchi con i numeri più alti.

66: ILLEGAL TRACK AND SECTOR
Il DOS ha cercato di accedere a una traccia o blocco che non esiste nel formato da utilizzare. Questo può indicare un problema di lettura del puntatore sul blocco successivo.

67: ILLEGAL SYSTEM T OR S
Questo particolare messaggio d'errore indica una traccia o blocco illegale del sistema.

70: NO CHANNEL (available)
Il canale richiesto non è accessibile o tutti i canali sono in uso. Con il DOS si possono aprire contemporaneamente un massimo di cinque file consecutivi. I canali ad accesso diretto possono avere sei file aperti.

71: DIRECTORY ERROR
La BAM non è uguale al calcolo interno. C'è un problema nell'assegnazione della BAM o la BAM è stata sovrascritta nella memoria del DOS. Per correggere questo problema, re-inizializzare il dischetto per ripristinare la BAM in memoria. Da questo intervento correttivo, alcuni file attivati potranno essere terminati.
NOTA: BAM = Block Availabity Map (Mappa per la Disponibilità dei Blocchi).

72: DISK FULL
O i blocchi sul dischetto sono tutti usati oppure la cartella si trova al suo limite di voci. DISK FULL viene inviato quando sul 1541 rimangono disponibili due blocchi per permettere di chiudere il file in corso.

73: DOS MISMATCH (73, CBM DOS V2.6 1541)
I DOS 1 e 2 sono compatibili in lettura ma non in scrittura. I dischi possono essere intercambiabili in lettura con entrambi i DOS, ma un disco formattato con una versione non può essere scritto su un disco formattato con l'altra versione perché il formato è diverso. Questo errore viene visualizzato ogni volta che si tenti di scrivere su un disco che è stato formattato in un formato non-compatibile. (È disponibile una routine di utilità per assistere nella conversione da un formato all'altro). Questo messaggio può apparire anche dopo l'inizializzazione.

74: DRIVE NOT READY
È stato fatto un tentativo di accedere al singolo Gestore-Disco 1541 senza aver inserito alcun dischetto.

APPENDICE C

IL BASIC DEL COMMODORE 64

Questo manuale ti è stato dato come introduzione al linguaggio BASIC -- sufficiente per farti prendere conoscenza alla programmazione del computer e farti conoscere qualche vocabolo coinvolto. Questa appendice ti dà un elenco completo delle regole (SINTASSI) del BASIC del 64 con le descrizioni precise. Si prega di fare prove con questi comandi. Ricorda, immettendo programmi non puoi danneggiare il computer, ed il modo migliore per imparare il computer è la sperimentazione.
Questa appendice è suddivisa in settori in base ai diversi tipi di operazioni in BASIC. I settori comprendono:

1. **Variabili ed Operatori:** descrive i diversi tipi di variabili, i nomi validi per le variabili, e gli operatori logici ed aritmetici.
2. **Comandi:** descrive i comandi usati per lavorare con i programmi come l'editazione, la memorizzazione e la cancellazione.
3. **Istruzioni:** descrive le istruzioni del programma BASIC usate nelle righe numerate dei programmi.
4. **Funzioni:** descrive le funzioni di stringa, numeriche e di stampa.

VARIABILI

Per il BASIC il Commodore 64 utilizza tre tipi di variabili: numerico reale, numerico intero e stringa (alfanumerico). I nomi delle variabili possono comprendere un'unica lettera dell'alfabeto, una lettera seguita da un numero o due lettere. Una variabile intera viene indicata utilizzando il segno per cento (%) dopo il nome della variabile. Le variabili di stringa hanno il segno del dollaro ($) dopo il loro nome.

Esempi

Nomi di variabili numeriche reali:	A, A5, BZ
Nomi di variabili numeriche intere:	A%, A5%, BZ%
Nomi di variabili di stringa:	A$, A5$, BZ$

Le matrici sono elenchi di variabili che hanno tutte lo stesso nome, ma usano numeri, chiamati sottocodici, per indicare l'elemento della matrice. Le matrici vengono definite con l'istruzione DIM e possono contenere variabili in virgola mobile (reali), numeri interi o stringhe. Il nome della variabile della matrice è seguita da un paio di parentesi () che racchiudono il numero delle variabili nell'elenco.

A(7), BZ%(11), A$(50), PT(20,20)

NOTA: Alcuni nomi delle variabili sono riservati all'utilizzo del 64 e non possono essere definite. Queste variabili sono: ST, TI e TI$. ST è una variabile di stato che relaziona le operazioni di entrata/uscita. Se ci fosse un problema di caricamento da disco o da nastro, verrà cambiato il valore di ST.
TI e TI$ sono variabili utilizzate per l'orologio in tempo reale del 64. La variabile TI viene aggiornata ogni sessantesimo di secondo. Essa inizia da 0 non appena viene acceso il computer e viene azzerata solo modificando il valore di TI$.
TI$ è una stringa che viene continuamente aggiornata dal sistema. I primi due caratteri corrispondono alle ore, il terzo e quarto caratteri ai minuti ed il quinto e sesto carattere sono i secondi. A questa variabile si può dare qualsiasi valore numerico, e verrà aggiornata da quel punto.

TI$ = "101530" imposta l'orologio sulle 10:15 e 30 secondi anti meridiani.

L'orologio viene cancellato quando si spenge il computer e riparte da zero non appena viene riacceso.

OPERATORI

Gli operatori aritmetici comprendono i seguenti segni:

+ Addizione (Somma)
- Sottrazione
* Moltiplicazione
/ Divisione
^ Elevazione a potenza (esponenziazione)

Su una riga che contega più di un operatore, questi ultimi rientrano sempre in un ordine prestabilito. Se sulla stessa riga vengono utilizzate insieme alcune operazioni, il computer assegnerà la precedenze come segue:
Prima di tutto l'esponenziazione, segue la moltiplicazione e la divisione e, da ultimo, l'addizione e la sottrazione.
Puoi cambiare l'ordine delle operazioni racchiudendo tra parentesi il calcolo da eseguirsi per primo. Le operazioni racchiuse fra parentesi verranno calcolate prima delle altre operazioni.
Ci sono anche operazioni per l'uguaglianza e l'ineguaglianza:

= Uguale a
< Minore di
> Maggiore di
<= Minore o uguale a
>= Maggiore o uguale a
<> Dioverso da

Infine ci sono tre operatori logici:

AND
OR
NOT

Questi ultimi vengono solitamente utilizzati per unire più formule nelle istruzioni IF...THEN. per esempio:

IF A=B AND C=D THEN 100 (Vero se entrambe le parti sono vere)
IF A=B OR C=D THEN 100 (Vero se una delle due parti è vera)

COMANDI

CONT (Continua)

Questo comando si utilizza per ricominciare l'esecuzione di un programma che è stato fermato o attraverso il tasto <STOP> (ma non <STOP> & <RESTORE>), o da un'istruzione STOP, oppure da un'istruzione END nel programma. Il programma riprenderà dal punto esatto in cui era stato fermato. CONT non funzionerà se hai cambiato o aggiunto righe al programma, o se il programma si è fermato a causa di un errore, o se hai provocato un errore prima di far ripartire il programma. In questi casi riceverai un errore CAN'T CONTINUE.

LIST (Lista)

Il comando LIST ti permette di esaminare le righe di un programma BASIC che si trovi in memoria. Puoi richiedere la visualizzazione di tutto il programma oppure solo di certi numeri di riga.

LIST	Mostra tutto il programma (se il programma è lungo riuscirai a vedere solo l'ultima schermata di righe)
LIST 10-	Mostra tutto il programma a partire dalla riga 10 (inclusa) (vedi sopra)
LIST 10	Mostra la sola riga 10 (se esiste)
LIST -10	Mostra il programma fino alla riga 10 (inclusa)
LIST 10-20	Mostra il programma compreso fra la riga 10 e la 20 (incluse)

LOAD (Carica)

Questo comando si utilizza per trasferire in memoria un programma, su nastro o su disco, in modo da poterlo utilizzare (eseguire). Digitando solo LOAD e premendo <RETURN>, in memoria verrà caricato il primo programma che si trova sull'unità a cassetta. Il comando può essere seguito da un nome di programma, racchiuso tra virgolette, al quale può seguire una virgola ed un numero, o variabile numerica, che agisce come numero di dispositivo ad indicare la provenienza del programma.
Se non viene aggiunto alcun numero per il dispositivo, il COMMODORE 64 assume il dispositivo #1, che è l'unità a cassetta. L'altro dispositivo comunemente usato con il comando LOAD è il dispositivo del disco, che riporta il numero #8.

LOAD	Legge il primo programma che capita su cassetta
LOAD "HELLO"	Cerca nella cassetta il programma chiamato HELLO e, una volta trovato, lo carica in memoria
LOAD A$	Cerca il programma il cui nome si trova nella variabile A$
LOAD "HELLO",8	Cerca sul disco il programma chiamato HELLO
LOAD "*",8	Cerca il primo programma sul disco

Se si vuole caricare un programma in codice macchina senza che venga riposizionato (su altro indirizzo della memoria), occorre aggiungere l'indirizzo secondario 1.

LOAD "M/C PROGRAM",1,1	Carica da nastro il codice macchina e lo posiziona allo stesso indirizzo indicato dal programma stesso

NEW (Nuovo)

Questo comando cancella tutto il programma in memoria e libera tutte le variabili che venivano utilizzate dallo stesso programma. A meno che il programma non sia stato salvato, verrà perso. Stai attento QUANDO USI QUESTO COMANDO.
Il comando NEW può essere utilizzato anche nei programmi come istruzione BASIC. Quando il programma raggiungerà questa riga, si auto-eliminerà. Questo metodo è utile quando, una volta eseguito il programma, si desidera lasciare la memoria libera.

RUN (Esegui)

Una volta che il programma è stato caricato in memoria, questo comando lo manda in esecuzione. Se di seguito al RUN non viene indicato nessun numero di riga, il computer inizierà l'esecuzione dal numero di riga più basso. Se invece è stato indicato un numero di riga, il programma inizierà l'esecuzione del programma da quel numero.

RUN	Inizia il programma dal numero di riga più basso
RUN 100	Inizia l'esecuzione dalla riga 100
RUN X	UNDEFINED STATEMENT ERROR. Con questo comando devi sempre indicare direttamente un numero di riga e non la sua rappresentazione in una variabile

SAVE (Salva)

Questo comando memorizzerà su cassetta o su disco il programma attualmente in memoria. Se si digita solo SAVE seguito da <RETURN>, il programma verrà salvato su cassetta. Poiché il computer non ha modo di sapere dell'esistenza di un eventuale programma già memorizzato su quel nastro, devi stare molto attento o potresti eliminare un programma molto utile. Se digiti SAVE seguito da un nome fra virgolette o da una variabile di stringa, il computer darà quel nome al programma, facendo in modo che possa risultare più facile individuarlo e recuperarlo in futuro. Il nome può essere seguito anche da un numero per il dispositivo. Dopo il numero del dispositivo, ci può essere una virgola ed un secondo numero: 0 o 1. Se questo secondo numero fosse 1, dopo aver salvato il tuo programma il COMMODORE 64 inserirà un marcatore EOT (END-OF-TAPE) per impedire che il computer possa guardare il nastro oltre quel marcatore in caso di un ulteriore comando LOAD. Infatti, cercando di caricare un programma oltre l'EOT, il computer risponderà con un errore FILE NON FOUND.

NOTA (per coloro un po' più esperti): Alla fine di ogni scrittura (salvataggio), il computer metterà sempre un marcatore per impedirgli di leggere oltre la fine del programma quando gli viene dato il comando LOAD. In questo caso, però, il marcatore è un EOF (END-OF-FILE) che, a differenza dell' EOT, non impedisce al computer di proseguire la ricerca di un eventuale programma, che si possa trovare dopo l'EOF, quando gli verrà dato il comando LOAD.

SAVE	Memorizza su nastro il programma senza nome
SAVE "HELLO"	Memorizza su nastro il programma chiamato HELLO
SAVE A$	Memorizza su nastro il programma il cui nome si trova nella variabile A$
SAVE "HELLO",8	Memorizza su disco con nome HELLO
SAVE "HELLO",1,1	Memorizza su nastro con nome HELLO e, con il prossi comando LOAD, lo ricarica nella stessa posizione della memoria
SAVE "HELLO",1,2	Memorizza su nastro con nome HELLO e fa seguire il programma con un marcatore di END-OF-TAPE
SAVE "HELLO",1,3	Come sopra ma con il successivo LOAD lo ricarica nella stessa posizione della memoria

VERIFY (Verifica)

Questo comando fa in modo che il computer controlli la validità del programma su disco o nastro con quello in memoria. Questa operazione prova che il programma sia stato realmente salvato nel caso che il nastro o il disco fossero stati difettosi o fosse successo qualche errore durante il salvataggio. VERIFY, senza nient'altro dopo il comando, fa in modo che il COMMODORE 64 controlli la validità del primo programma che si trova su nastro, indipendentemente dal nome, con il programma in memoria. VERIFY, seguito da un nome di programma o da una variabile di stringa, cercherà quel programma da controllare. Con il comando VERIFY si possono aggiungere anche numeri per i dispositivi.

VERIFY	Controlla il primo programma su nastro
VERIFY "HELLO"	Cerca HELLO e lo verifica con la memoria
VERIFY "HELLO",8	Cerca HELLO su disco, quindi lo verifica

Per controllare se un programma si trova già su un nastro, basta dare VERIFY ed il computer dirà quale programma ha trovato (se c'è).

ISTRUZIONI

CLOSE (Chiudi)

Questo comando completa e chiude qualsiasi file usati dalle istruzioni OPEN. Il numero che segue CLOSE è il numero del file da chiudere.

CLOSE 2	Viene chiuso solo il file #2

CLR (Clear - Vuota)

Questo comando cancellerà tutte le variabili in memoria, ma lascerà intatto il programma. Questo comando viene eseguito automaticamente quando si dà il comando RUN.

CMD (Comanda)

CMD direziona l'uscita, che di solito andrebbe allo schermo (cioè, le istruzioni PRINT, LIST, ma non le POKE date sullo schermo), verso un altro dispositivo che potrebbe essere una stampante o un file di dati su disco.
Detto dispositivo o file deve prima essere aperto. Il comando CMD deve essere seguito da un numero o da una variabile numerica che si riferisca al file.

OPEN 1,4	Apre il dispositivo #4, che è la stampante
CMD 1	Tutte le normali uscite sono direzionate sulla stampante
LIST	Adesso il listato del programma va sulla stampante e non sullo schermo

Per ridirezionare l'uscita verso lo schermo, chiudi il file con CLOSE 1.

DATA

Questa istruzione è seguita da un elenco di voci da utilizzare con le istruzioni READ. Le voci possono essere valori numerici o stringhe di testo, entrambi separate da virgole. Gli elementi di stringa non devono essere fra virgolette, a meno che non contengano spazi, due punti, od altre virgole. Se tra due o più virgole non si trova alcun elemento, il valore verrà considerato come uno zero in caso di valore numerico, oppure come una stringa vuota.

DATA 12, 14.5, "HELLO, MOM", 3.14, PART1

DEF FN (Definisci funzione)

Questo comando ti permette di definire un calcolo complesso come fosse una funzione con nome corto. Nel caso di una formula lunga che venga utilizzata molte volte dal programma, questo comando può far guadagnare tempo e spazio. Il nome della funzione sarà FN e qualsiasi nome valido per la variabile (lungo 1 o 2 caratteri). Prima di tutto devi definire la funzione utilizzando l'istruzione DEF seguita dal nome della funzione. A seguito del nome, una serie di parentesi che racchiudono una variabile numerica. Quindi segue la vera formula che vuoi definire, con la variabile nel posto giusto. In seguito potrai "richiamare" la formula sostituendo la variabile con qualsiasi numero.

```
10 DEF FN A(X) = 12 * ( 34.75 - X / .3 )
20 PRINT FN A(7)                 ^
            ^                    |
            |                    |   il 7 viene inserito dove
            +--------------------+   nella formula si trova X
```

In questo esempio il risultato sarà 137.

DIM (Dimensiona)

Quando in una matrice ti servono più di 11 elementi, devi eseguire un istruzione DIM. Ricorda che tutta la matrice occupa spazio in memoria, per cui non crearne una che sia più grande del necessario. Per conoscere la quantità di variabili create con DIM, moltiplica la quantità totale degli elementi, più uno, per ciascuna dimensione della matrice.

```
10 DIM A$(40), B7(15), CC%(4,4,4)
         ^        ^          ^
         |        |          |
41 elementi  16 elementi  125 elementi
```

Con l'istruzione DIM puoi dimensionare più di una matrice. Comunque sia, stai attento a non dimensionare più di una volta una stessa matrice.

END (Fine)

Quando un programma incontra una istruzione END, il programma si ferma come se avesse esaurito le righe. Per far ripartire il programma da quel punto (nel caso non fosse l'ultima riga del programma), puoi usare CONT.

FOR ... TO ... STEP (Per… A… Passo)

Questa istruzione lavora con l'istruzione NEXT e ripete una parte del programma una quantità di volte prestabilita. Il formato è:

FOR (nome della variabile)=(inizio conteggio) TO (fine conteggio) STEP (con passi di)

La variabile della sequenza verrà aggiunta o sottratta durante il programma. Senza l'indicazione di STEP, la somma o la sottrazione alla variabile sarà di 1. Inizio conteggio e fine conteggio sono i limiti che assumerà il valore della variabile nella sequenza.

```
10 FOR L = 1 TO 10 STEP .1
20 PRINT L
30 NEXT L
```

La fine del valore della sequenza può essere seguito dal comando STEP ed un altro numero o variabile. In questo caso, invece di 1, ogni volta viene aggiunto il valore che segue STEP. Questo ti permette di contare all'indietro o per frazione.

GET (Ricevi da tastiera)

L'istruzione GET ti permette di 'catturare' dati dalla tastiera, un carattere alla volta. Quando GET viene eseguito, il carattere digitato viene assegnato alla variabile. Se nel momento dell'esecuzione di GET non viene digitato alcun carattere, alla variabile verrà assegnato un carattere nullo (vuoto). GET è seguito da un nome di variabile, di solito una variabile di stringa. Se venisse usata una variabile numerica, alla pressione di un tasto alfanumerico il programma si fermerebbe con un messaggio d'errore. L'istruzione GET può essere messa in una sequenza che controlli qualsiasi risultato non avvenuto. Questa sequenza continuerà a controllarsi fino a quando non venga premuto un tasto (vedi l'esempio qui sotto).

```
10 GET A$: IF A$ = "" THEN 10
```

GET# (Ricevi da file o dispositivo)

L'istruzione GET# viene utilizzata, con un dispositivo o file precedentemente aperto, per inserire un carattere alla volta da quel dispositivo o file.

GET#1,A$

Questo comando inserirà un carattere da un file di dati.

GOSUB (Salta alla subroutine...)

Questa istruzione è simile al GOTO, tranne che il computer si ricorda qual è stata l'ultima riga del programma che ha eseguito prima del GOSUB. Al momento che incontra una riga con l'istruzione RETURN, il programma ritorna all'istruzione che seguiva il GOSUB. Questo sistema è utile quando nel tuo programma si trova una routine alla quale devi ricorre più volte. Invece di immettere più volte la stessa routine, basta eseguire il GOSUBs ogni volta che diventa necessaria la routine.

```
20 GOSUB 800
```

GOTO o **GO TO** (Vai a...)

Quando viene raggiunta un'istruzione con il comando GOTO, la riga successiva da eseguire sarà quella con il numero di riga che segue la parola GOTO.

IF ... THEN (Se... Allora)

IF... THEN permette al computer di analizzare una situazione e, in base al risultato, di prendere due percorsi possibili di processo. Se l'espressione risulta vera, verrà eseguita l'istruzione che segue il THEN e che potrà essere qualsiasi istruzione BASIC.
Se l'espressione risulta falsa, il programma passa direttamente alla riga successiva. L'espressione da valutare può essere una variabile o una formula, in tal caso sarà considerata vera se nonzero, e falsa se zero. Nella maggior parte dei casi è implicata un'espressione con operatori relationali (=, <, >, <=, >=, <>, AND, OR, NOT).

```
10 IF X > 10 THEN END       Se X è maggiore di 10. termina
```

INPUT (Aspetta il dato da tastiera)

L'istruzione INPUT permette al programma di ricevere un dato dall'utente per poi assegnarlo ad una variabile. Il programma si fermerà, stampando un punto interrogativo (?) sullo schermo, e rimarrà in attesa che l'utente immetta la risposta e prema <RETURN>.
INPUT è seguito da un nome di variabile o un elenco di nomi di variabili separati da virgole. L'istruzione INPUT, prima dell'elenco dei nomi delle variabili, supporta anche un eventuale messaggio da inserire all'interno di virgolette che ne delimitano l'inizio e la fine. Se le risposte devono servire per più variabili, quando digitate devono essere separate da virgole.

```
10 INPUT "INSERISCI IL TUO PRIMO NOME";A$
20 PRINT "INSERISCI IL NUMERO DEL TUO CODICE;: INPUT B
```

INPUT# (Prendi il dato da dispositivo o file)

INPUT# è simile a INPUT, ma prende dati da un file precedentemente aperto con OPEN.

```
10 INPUT#1, A
```

LET (Assegna)

LET nei programmi non viene quasi mai utilizzato, perché facoltativo, ma questa istruzione è il cuore di tutti i programmi BASIC. Il nome della variabile, alla quale viene assegnato il risultato di un calcolo, si trova sul lato sinistro del segno uguale, la formula sul lato destro.

```
10 LET A = 5              come 10 a = 5
20 LET D$ = "HELLO"       come 20 D$ 0 “HELLO”
```

NEXT (Salta alla prossima istruzione)

NEXT viene utilizzato sempre insieme all'istruzione FOR. Quando il programma raggiunge una istruzione NEXT, esso controlla l'istruzione FOR per vedere se è stato raggiunto il limite della variabile di ciclo. Se il ciclo non è finito, la variabile di ciclo viene incrementata dal valore indicato da STEP ed il programma riesegue tutte le istruzioni comprese tra il FOR ed il NEXT. Se il ciclo è finito, l'esecuzione passa all'istruzione che segue il NEXT. NEXT può essere seguito da un nome di variabile o da un elenco di nomi di variabili separati da virgole. Nel caso in cui non venga indicata nessun nome di variabile, viene portato avanti l'ultimo ciclo iniziato. Se invece qui si trovano più nomi di variabili, vengono completati i cicli nell'ordine da sinistra a destra.

```
10 FOR X = 1 TO 100: NEXT
```

ON (In base a... Vai su...)

Questo comando trasforma i comandi GOTO e GOSUB in particolari versioni dell'istruzione IF. ON è seguito da una formula (o da una variabile) che viene valutata. Se il risultato del calcolo è 1, il programma salterà alla prima riga dell'elenco; se il risultato è 2 il programma salterà alla seconda riga seconda dell'elenco, e così via. Se il risultato è 0, negativo, o superiore alla quantità dei numeri riportati nell'elenco, verrà eseguita l'istruzione sulla riga successiva che segue l'istruzione ON.

```
10 INPUT X
20 ON X GOTO 10,20,30,40,50
```

OPEN (Apri)

L'istruzione OPEN permette al COMMODORE 64 di accedere ai dispositivi come il disco per i dati, una stampante o anche lo schermo. OPEN è seguito da un numero (0-255) con il quale si riferiranno tutte le istruzioni che seguiranno. Di solito, dopo il primo, c'è un secondo numero che rappresenta il dispositivo. I numeri dei dispositivi sono:

0 Schermo
1 Cassetta
4 Stampante
8 Disco

A seguito del numero del dispositivo può esserci un terzo numero, nuovamente separato da una virgola, che è l'indirizzo secondario. Nel caso del disco, il numero si riferisce al numero del buffer, o canale. Nella stampante l'indirizzo secondario controlla caratteristiche come la stampa espansa. Per ulteriori dettagli consulta il Manuale di Riferimento del Programmatore del Commodore 64.

OPEN 1,0	Apre lo schermo come dispositivo
OPEN 2,8,8,"D"	Apre il disco in lettura; il file da cercare è D

OPEN 3,4 Apre la stampante
OPEN 4,8,15 Apre il canale dei dati sul disco

Vedi anche: CLOSE, CMD, GET#, INPUT# e PRINT#, la variabile di sistema ST e l'appendice B.

POKE (Inserisci)

POKE è sempre seguito da due numeri o formule. La prima posizione è una locazione di memoria; il secondo numero è un valore decimale, da 0 al 255, che verrà inserito nella locazione di memoria sostituendo qualsiasi altro valore precedentemente memorizzato.

```
10 POKE 53281,0
20 S = 4096 * 13
30 POKE S + 29,8
```

PRINT (Stampa su schermo)

L'istruzione PRINT è la prima che viene utilizzata dalla maggior parte delle persone, ma occorre fare attenzione alla quantità di variazioni che ci sono. PRINT può essere seguito da:

Stringa di Testo fra virgolette
Nomi di Variabili
Funzioni
Segni di punteggiatura

I segni di punteggiatura vengono utilizzati per formattare i dati sullo schermo. La virgola suddivide lo schermo in quattro colonne, mentre il punto e virgola sopprime tutta la spaziatura. Entrambi i segni possono essere messi come ultimo simbolo di una riga. In questo caso la stampa successiva avverrà come fosse una continuazione della stessa istruzione PRINT.

```
10 PRINT "HELLO"
20 PRINT "HELLO", A$
30 PRINT A + B
40 PRINT J;
50 PRINT A,B,C,D
```

Vedi anche: le funzioni POS, SPC e TAB.

PRINT# (Stampa su dispositivo o file)

Tra questa istruzione e PRINT ci sono alcune differenze. PRINT# è seguito da un numero che si riferisce al dispositivo o file di dati precedentemente aperto. Detto numero è seguito da una virgola e da un elenco da stampare. La virgola ed il punto e virgola hanno lo stesso effetto del PRINT. Da notare che alcuni dispositivi non possono lavorare con TAB e SPC.

```
100 PRINT#1, "DATA VALUES"; A%, B1, C$
```

READ (Leggi)

READ viene utilizzato per assegnare a varibili le informazioni lette dalle istruzioni DATA, e fare in modo che le informazioni possano essere utilizzate. Particolare attenzione deve essere fatta per evitare la lettura di stringhe quando READ si aspetta un numero; in questo caso verrà dato un errore TYPE MISMATCH.

REM (Commento)

REM non è altro che una nota per permettere a chiunque di leggere una fase del programma. Con questo si può spiegare una parte del programma o dare ulteriori istruzioni. Le istruzioni REM non influiscono in alcun modo sullo svolgimento del programma, se non farne aumentare la lunghezza. REM può essere seguito da qualsiasi testo.

RESTORE (Ripristina)

Quando eseguito in un programma, il puntatore che punta la voce da leggere (con l'istruzione READ) nell'elenco dei DATA verrà riposizionato alla prima voce dell'elenco. Questo ti dà la possibilità di rileggere l'informazione. RESTORE sta da solo su una riga.

RETURN (Ritorna)

Questa istruzione viene utilizzata sempre insieme a GOSUB. Quando il programma incontra un RETURN, salterà all'istruzione che segue immediatamente il comando GOSUB. Nel caso in cui non sia stato precedentemente inserito nessun GOSUB, capiterà un errore RETURN WITHOUT GOSUB.

STOP (Fermati)

Questa istruzione arresterà l'esecuzione del programma. In questo caso verrà visualizzato il messaggio BREAK IN xxx, dove xxx è il numero di riga che contiene lo STOP. Il programma può essere fatto riproseguire con il comando CONT. Di solito STOP viene utilizzato durante la depurazione di un programma.

SYS (Parti dalla locazione...)

SYS è seguito da un numero decimale o valore numerico compreso da 0 a 65535. In questo caso il programma inizierà l'esecuzione del programma in linguaggio macchina a partire da quella locazione di memoria. Il comando SYS è simile alla funzione USR, ma non permette il passaggio di parametri.

WAIT (Aspetta)

WAIT viene utilizzato per fermare il programma fino a quando non cambia, in un modo ben preciso, il contenuto di una locazione di memoria. WAIT è seguito da una locazione della memoria (X) e fino a due variabili. Il formato è:

WAIT X,Y,Z

Il contenuto della locazione della memoria viene prima sottoposto all'OR esclusivo con il terzo numero, se presente, ed in seguito una valutazione AND con il secondo numero. Se il risultato è zero, il programma ritorna a quella locazione di memoria e controlla un'altra volta. Non appena il risultato diventa nonzero, il programma continua con l'istruzione successiva.

FUNZIONI NUMERICHE

ABS(X) (valore assoluto)

ABS ritorna il valore assoluto del numero, senza il suo segno (+ o -). La risposta è sempre positiva.

ATN(X) (arcotangente)

Ritorna l'angolo, misurato in radianti, la cui tangente è X.

COS(X) (coseno)

Ritorna il valore del coseno di X, dove X è un angolo misurato in radianti.

EXP(X) (esponenziale)

Ritorna il valore della costante matematica *e* (2.71828183) elevata alla potenza di X.

FN xx(X) (funzione)

Ritorna il valore della funzione xx definita dall'utente e creata in una istruzione DEF FN xx(X).

INT(X) (intero)

Ritorna il valore troncato di X, cioè senza tutte le posizioni decimali a destra del punto decimale. Il risultato sarà sempre minore di, o uguale a, X. In questo modo, qualsiasi numero negativo, con posizioni decimali, diventerà l'intero minore del loro valore attuale.

LOG(X) (logaritmo)

Ritornerà il logaritmo naturale di X. Il logaritmo naturale di base *e* (vedi EXP(X)). Per convertire al logaritmo di base 10, basta dividere per LOG(10).

PEEK(X) (Guarda)

Usato per scoprire il contenuto della locazione di memoria X, nel campo da 0 a 65535, dà un risultato da 0 a 255. PEEK è spesso usato insieme all'istruzione POKE.

RND(X) (numero casuale)

RND(X) ritorna un numero casuale tra 0 ed 1. Il primo numero casuale dovrà essere generato dalla formula RND(-TI) per fare in modo che ogni volta sia differente. Dopo questo, X dovrà essere un 1 o un qualsiasi numero positivo. Se X è zero, il risultato sarà lo stesso dell'ultimo numero casuale. Un valore negativo di X resetterà il generatore. L'uso dello stesso numero negativo per X porterà alla medesima sequenza di numeri "casuali".
La formula per la generazione di un numero tra X e Y è:

N = RND(1) * (Y-X) + X

dove,
Y è il valore superiore,
X è il valore inferiore dei numeri richiesti.

SGN(X) (segno)

Questa funzione ritorna il segno (positivo, negativo o zero) di X. Il risultato sarà +1 se positivo, 0 se zero e -1 se negativo.

SIN(X) (seno)

SIN(X) è la funzione del seno trigonometrico. Il risultato sarà il seno di X, dove X è un angolo in radianti.

SQR(X) (radice quadrata)

Questa funzione ritornerà la radice quadrato di X, dove X è un numero positivo oppure 0. Se X fosse negativo, ne risulterà un ILLEGAL QUANTITY ERRORE (ERRORE ILLEGALE).

TAN(X) (tangente)

Il risultato sarà la tangente di X, dove X è un angolo in radianti.

USR(X)

Quando si utilizza questa funzione, il programma salterà ad un programma in linguaggio macchina il cui punto iniziale è contenuto nelle locazioni di memoria. Il parametro X viene passato al programma in linguaggio macchina e questo ritornerà un altro valore al programma BASIC. Per ulteriori particolari su questa funzione, fare riferimento al Commodore 64 Programmer's Reference Manual ed alla programmazione in linguaggio macchina.

FUNZIONI DI STRINGA

ASC(X$)

Questa funzione ritornerà il codice ASCII del primo carattere di X$.

CHR$(X)

Questa funzione è l'opposto di ASC e ritorna un carattere di stringa il cui codice ASCII è X.

**LEFT$**(X$,X)

Ritorna i caratteri più a sinistra di X$, partendo dal carattere X esimo.

LEN(X$)

Il risultato sarà la quantità di caratteri (inclusi gli spazi ed altri simboli) della stringa X$.

**MID$**(X$,S,X)

La funzione ritornerà una stringa contenente X caratteri, iniziando dal carattere 'S'esimo, di X$.

**RIGHT$**(X$,X)

Ritorna i caratteri più a destra di X$, partendo dal carattere X esimo.

STR$(X)

Questa ritornerà una stringa identica alla versione stampata (PRINT) di X.

VAL(X$)

Questa funzione converte X$ in un numero, ed essenzialmente è l'operazione inversa di STR$. La stringa viene esaminata iniziando dal carattere più a sinistra e procedendo verso destra, con tutti i caratteri che siano riconosciuti con il formato-numero.

```
10 X = VAL("123.456")           X = 123.456
10 X = VAL("12A13B")            X = 12
10 X = VAL("RIU017")            X = 0
10 X = VAL("-1.23.45.67")       X = -1.23
```

ALTRE FUNZIONI

FRE(X)

Questa funzione ritorna la quantità dei byte non utilizzati, disponibili in memoria, indipendentemente dal valore di X. Da notare che, in caso di byte non usati sopra i 32 Kb, FRE(X) riporterà numeri negativi.

POS(X)

Questa funzione ritorna il numero della colonna (0-39) dalla quale inizierà la successiva istruzione PRINT sullo schermo. X può avere un valore qualsiasi e non viene utilizzato.

SPC(X)

Questa si utilizza in una istruzione PRINT per saltare X spazi in avanti.

TAB(X)

Anche TAB viene usato in una istruzione PRINT; la voce successiva da stampare si troverà nella colonna X.

APPENDICE D

ABBREVIAZIONI PER LE PAROLE-CHIAVE DEL BASIC

Per risparmiare tempo durante l'immissione dei programmi e dei comandi, il BASIC del Commodore 64 permette all'utente di abbreviare la maggior parte delle parole chiave. L'abbreviazione per il PRINT è un punto interrogativo. L'abbreviazione per le altre parole viene eseguito immettendo il primo, o i primi due caratteri della parola, seguito dalla lettera successiva della parola insieme al tasto SHIFT. Se si usano le abbreviazioni in una riga del programma, con il comando LIST le parole chiave appariranno nella loro forma completa.

Comamdo	Abbreviazione	Cosa si vede sullo schermo	Comando	Abbreviazione	Cosa si vede sullo schermo
ABS	A <SHIFT+N>		NOT	N <SHIFT+O>	
AND	A <SHIFT+N>		ON	------	ON
ASC	A <SHIFT+S>		OPEN	O <SHIFT+P>	
ATN	A <SHIFT+T>		OR	------	OR
CHR$	C <SHIFT+H>		PEEK	P <SHIFT+E>	
CLOSE	CL <SHIFT+O>		POKE	P <SHIFT+O>	
CLR	C <SHIFT+L>		POS	------	POS
CMD	C <SHIFT+M>		PRINT	?	?
CONT	C <SHIFT+O>		PRINT#	P <SHIFT+R>	
COS	------	COS	READ	R <SHIFT+E>	
DATA	D <SHIFT+A>		REM	------	REM
DEF	D <SHIFT+E>		RESTORE	RE <SHIFT+S>	
DIM	D <SHIFT+I>		RETURN	RE <SHIFT+T>	
END	E <SHIFT+N>		RIGHT$	R <SHIFT+I>	
EXP	E <SHIFT+X>		RND	R <SHIFT+N>	
FN	------	FN	RUN	R <SHIFT+U>	
FOR	F <SHIFT+O		SAVE	S <SHIFT+A>	
FRE	F <SHIFT+R>		SGN	S <SHIFT+G>	
GET	G <SHIFT+E>		SIN	S <SHIFT+I>	
GET#	------	GET#	SPC(	S <SHIFT+P>	
GOSUB	GO <SHIFT+S>		SQR	S <SHIFT+Q>	
GOTO	G <SHIFT+O>		STATUS	ST	ST
IF	------	IF	STEP	ST <SHIFT+E>	
INPUT	------	INPUT	STOP	S <SHIFT+T>	
INPUT#	I <SHIFT+N>		STR$	ST <SHIFT+R>	
INT	------	INT	SYS	S <SHIFT+Y>	
LEFT$	LE <SHIFT+F>		TAB(	T <SHIFT+A>	
LEN	------	LEN	TAN	------	TAN
LET	L <SHIFT+E>		THEN	T <SHIFT+H>	
LIST	L <SHIFT+I>		TIME	TI	TI
LOAD	L <SHIFT+O>		TIME$	TI$	TI$
LOG	------	LOG	USR	U <SHIFT+S>	
MID$	M <SHIFT+I>		VAL	V <SHIFT+A>	
NEW	------	NEW	VERIFY	V <SHIFT+E>	
NEXT	N <SHIFT+E>		WAIT	W <SHIFT+A>	

NOTA: nelle due colonne “Cosa si vede sullo schermo” dovrebbero essere riportati I simboli grafici che presenta il Commodore digitando la combinazione dei tasti interessata. Purtroppo, salvo creandoli con qualche programma di grafica, non sono disponibili in nessun editore di testi.

APPENDICE E

CODICI DI VISUALIZZAZIONE SU SCHERMO

La seguente tabella elenca tutti i caratteri compresi nella serie dei caratteri del Commodore 64. Essa mostra quali numeri dovranno essere inseriti (POKE) nella memoria di schermo (locazioni 1024-2023) per ottenere un dato carattere. Inoltre mostra a quale carattere corrisponde un numero prelevato (PEEK) dallo schermo.
Sono disponibili due serie di caratteri, ma solo una serie alla volta. Questo significa che sullo schermo non è possibile visualizzare caratteri di una serie contemporaneamente a caratteri dell'altra serie. Le serie si scambiano premendo contemporaneamente i tasti <SHIFT> e <C=>.
Attraverso il BASIC, con POKE 53272,21 si passerà alla modalità maiuscola e con POKE 53272,23 alla minuscola.
Ogni numero sulla tabella può essere visualizzato anche in modalità INVERSA. Detto carattere si può ottenere aggiungendo 128 ai valori mostrati.
Se vuoi visualizzare un cerchietto solido nella locazione 1504, inserisci (POKE) il codice del cerchio (81) nella locazione 1504: POKE 1504,81.
Qui c'è una corrispondente locazione di memoria per controllare il colore di ciascun carattere visualizzato sullo schermo (locazioni 55296-56295). Per cambiare in giallo (codice colore 7) al cerchietto solido, dovrai inserire (POKE) il colore del carattere nella corrispondente locazione di memoria: POKE 55776,7.
Per le mappe di memoria dello schermo e del colore, con i codici del colore, fai riferimento alla Appendice G.

CODICI DI SCHERMO

Serie 1	Serie 2	POKE		Serie 1	Serie 2	POKE		Serie 1	Serie 2	POKE
@		0		+		43			V	86
A	a	1		,		44			W	87
B	b	2		-		45			X	88
C	c	3		.		46			Y	89
D	d	4		/		47			Z	90
E	e	5		0		48				91
F	f	6		1		49				92
G	g	7		2		50				93
H	h	8		3		51				94
I	i	9		4		52				95
J	j	10		5		53				96
K	k	11		6		54				97
L	l	12		7		55				98
M	m	13		8		56				99
N	n	14		9		57				100
O	o	15		:		58				101
P	p	16		;		59				102
Q	q	17		<		60				103
R	r	18		=		61				104
S	s	19		>		62				105
T	t	20		?		63				106
U	u	21				64				107
V	v	22			A	65				108
W	w	23			B	65				109
X	x	24			C	67				110
Y	y	25			D	68				111
Z	z	26			E	69				112
[		27			F	70				113
£		28			G	71				114
]		29			H	72				115
^		30			I	73				116
<-		31			J	74				117
SPAZIO		32			K	75				118
!		33			L	76				119
"		34			M	77				120
#		35			N	78				121
$		36			O	79				122
%		37			P	80				123
&		38			Q	81				124
'		39			R	82				125
(		40			S	83				127
)		41			T	84				
*		42			U	85				

I codici da 128 a 255 sono le immagini invertire dei codici da 0 a 127.

APPENDICE F

CODICI ASCII E CHR$

Questa appendice ti mostra quali caratteri appariranno con il comando PRINT CHR$ (X), con tutti i valori possibili di X. Essa mostrerà anche i valori ottenuti immettendo PRINT ASC("x"), dove x è un qualsiasi carattere che puoi digitare. La tabella è utile per valutare il carattere ricevuto in una istruzione GET, per convertire in maiuscolo/minuscolo e per stampare comandi per i caratteri (come passare in maiuscolo/minuscolo) che non siano racchiusi tra virgolette.

Stampa	CHR$	Stampa	CHR$	Stampa	CHR$	Stampa	CHR$
	0	0	48		96	-nero-	144
	1	1	49		97	-su-	145
	2	2	50		98	-rvs off-	146
	3	3	51		99	-clr-	147
	4	4	52		100	-inst-	148
-bianco-	5	5	53		101	-marrone-	149
	6	6	54		102	-rosso chiaro-	150
	7	7	55		103	-grigio 1-	151
SHIFT+C=off	8	8	56		104	-grigio 2-	152
SHIFT+C=on	9	9	57		105	-verde chiaro-	153
	10	:	58		106	-blu chiaro-	154
	11	;	59		107	-grigio 3-	155
	12	<	60		108	-viola-	156
return	13	=	61		109	-sinistra-	157
minuscolo	14	>	62		110	-giallo-	158
	15	?	63		111	-ciano-	159
	16	@	64		112	SPAZIO	160
-giù-	17	A	65		113		161
-rvs on-	18	B	66		114		162
-home-	19	C	67		115		163
-del-	20	D	68		116		164
	21	E	69		117		165
	22	F	70		118		166
	23	G	71		119		167
	24	H	72		120		168
	25	I	73		121		169
	26	J	74		122		170
	27	K	75		123		171
-rosso-	28	L	76		124		172
-destra-	29	M	77		125		173
-verde-	30	N	78		126		174
-blu-	31	O	79		127		175
SPAZIO	32	P	80		128		176
!	33	Q	81	-arancio-	129		177
"	34	R	82		130		178
#	35	S	83		131		179
$	36	T	84		132		180
%	37	U	85	f1	133		181
&	38	V	86	f3	134		182
'	39	W	87	f5	135		183
(	40	X	88	f7	136		184
)	41	Y	89	f2	137		185
*	42	Z	90	f4	138		186
+	43	[	91	f6	139		187
,	44	£	92	f8	140		188
-	45	]	93	SHIFT+RETURN	141		189
.	46	^	94	maiuscolo	142		190
/	47	-freccia- - sinistra-	95				

I CODICI 192-223 SONO UGUALI A 96-127
I CODICI 224-254 SONO UGUALI A 160-190
IL CODICE 255 È UGUALE A 126

APPENDICE G

MAPPE DELLA MEMORIA DI SCHERMO E COLORE

Le tabelle che seguono elencano le locazioni di memoria che controllano l'inserimento dei caratteri sullo schermo e le locazioni usate per cambiare colore al singolo carattere, ma mostrando anche i codici di colore del carattere.

MAPPA DELLA MEMORIA DI SCHERMO

```
                            COLONNA                          1063
      0             10             20             30             39 /
     +------------------------------------------------------------/
1024 |                                                            |  0
1064 |                                                            |
1104 |                                                            |
1144 |                                                            |
1184 |                                                            |
1224 |                                                            |
1264 |                                                            |
1304 |                                                            |
1344 |                                                            |
1384 |                                                            |
1424 |                                                            | 10
1464 |                                                            |
1504 |                                                            |   RIGA
1544 |                                                            |
1584 |                                                            |
1624 |                                                            |
1664 |                                                            |
1704 |                                                            |
1744 |                                                            |
1784 |                                                            |
1824 |                                                            | 20
1864 |                                                            |
1904 |                                                            |
1944 |                                                            |
1984 |                                                            | 24
     +------------------------------------------------------------\
                                                                   \
                                                                  2023
```

I valori attuali da inserire (POKE) in una locazione di memoria del colore, per cambiare un colore al carattere, sono:

0 NERO	4 VIOLA	8 ARANCIO	12 GRIGIO 2
1 BIANCO	5 VERDE	9 MARRONE	13 VERDE CHIARO
2 ROSSO	6 BLU	10 ROSSO CHIARO	14 BLU CHIARO
3 CIANO	7 GIALLO	11 GRIGIO 1	GRIGIO 3

Per esempio: per cambiare in rosso un carattere situato nell'angolo sinistro superiore dello schermo, digitare: POKE 55296,2.

MAPPA DELLA MEMORIA PER IL COLORE

```
                                   COLONNA                            55335
       0             10              20              30             39 /
     +----------------------------------------------------------------/
55296|                                                                |  0
55336|                                                                |
55376|                                                                |
55416|                                                                |
55456|                                                                |
55496|                                                                |
55536|                                                                |
55576|                                                                |
55616|                                                                |
55656|                                                                |
55696|                                                                | 10
55736|                                                                |
55776|                                                                |   RIGA
55816|                                                                |
55856|                                                                |
55896|                                                                |
55936|                                                                |
55976|                                                                |
56016|                                                                |
56056|                                                                |
56096|                                                                | 20
56136|                                                                |
56176|                                                                |
56216|                                                                |
56256|                                                                | 24
     +----------------------------------------------------------------\
                                                                       56295
```

APPENDICE H

FUNZIONI MATEMATICHE DERIVATE

Le funzioni che non sono essenziali al BASIC del Commodore, possono essere calcolate come segue:

Funzione	Equivalente del BASIC
SECANTE	SEC(X)=1/COS(X)
COSECANTE	CSC(X)=1/SIN(X)
COTAMGENTE	COT(X)=1/TAN(X)
SENO INVERSO	ARCSIN(X)=ATN(X/SQR(-X*X+1))
COSENO INVERSO	ARCCOS(X)=-ATN(X/SQR(-X*X+1))+{µ}/2
SECANTE INVERSA	ARCSEC(X)=ATN(X/SQR(X*X-1))
COSECANTE INVERSA	ARCCSC(X)=ATN(X/SQR(X*X-1)) +(SGN(X)-1*{µ}/2
COTANGENTE INVERSA	ARCOT(X)=ATN(X)+{pi}/2
SENO IPERBOLICO	SINH(X)=(EXP(X)-EXP(-X))/2
COSENO IPERBOLICO	COSH(X)=(EXP(X)+EXP(-X))/2
TANGENTE IPERBOLICA	TANH(X)=EXP(-X)/(EXP(X)+EXP(-X))*2+1
SECANTE IPERBOLICA	SECH(X)=2/(EXP(X)+EXP(-X))
COSECANTE IPERBOLICA	CSCH(X)=2/(EXP(X)-EXP(-X))
COTANGENTE IPERBOLICA	COTH(X)=EXP(-X)/(EXP(X)-EXP(-X))*2+1
SENO IPERBOLICO INVERSO	ARCSINH(X)=LOG(X+SQR(X*X+1))
COSENO IPERBOLICO INVERSO	ARCCOSH(X)=LOG(X+SQR(X*X-1))
TANGENTE IPERBOLICA INVERSA	ARCTANH(X)=LOG((1+X)/(1-X))/2
SECANTE IPERBOLICA INVERSA	ARCSECH(X)=LOG((SQR(-X*X+1)+1/X)
COSECANTE IPERBOLICA INVERSA	ARCCSCH(X)=LOG((SGN(X)*SQR(X*X+1/X)
COTANGENTE IPERBOLICA INVERSA	ARCCOTH(X)=LOG((X+1)/(X-1))/2

APPENDICE I

PIEDINATURA DEI DISPOSITIVI DI ENTRATA/USCITA

Questa appendice si propone di mostrarti quali possono essere i collegamenti al Commodore 64.

1) I/O giochi
2) Postazione cartuccia
3) Audio/Video
4) I/O seriale (Lettore dischi/Stampante)
5) Uscita modulatore
6) Lettore di cassette
7) Porta utente

Porta di controllo 1

Pin	Tipo	Nota
1	JOYA0	
2	JOYA1	
3	JOYA2	
4	JOYA3	
5	POT AY	
6	PULSANTE A/LP	
7	+5V	MAX 100mA
8	GND	
9	POT AX	

Porta di controllo 2

Pin	Tipo	Nota
1	JOYB0	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY	
6	PULSANTE B	
7	+5V	MAX 100mA
8	GND	
9	POT BX	

```
/---------------------\
|   1    2    3    4    5   |
|   O    O    O    O    O   |
 |                          |
 |     O    O    O    O     |
  |    6    7    8    9    |
   \_______________/
```

Porta di espansione per Cartuccia

Pin	Tipo	Pin	Tipo	Pin	Tipo	Pin	Tipo
1	GND	12	BA	A	GND	N	A9
2	+5V	13	/DMA	B	/ROMH	P	A8
3	+5V	14	D7	C	/RESET	R	A7
4	/IRQ	15	D6	D	/NMI	S	A6
5	R/W	16	D5	E	02	T	A5

6	Dot Clock	17	D4	F	A15	U	A4
7	I/O1	18	D3	H	A14	V	A3
8	/GAME	19	D2	J	A13	W	A2
9	/EXROM	20	D1	K	A12	X	A1
10	I/O2	21	D0	L	A11	Y	A0
11	/ROML	22	GND	M	A10	Z	GND

```
     2 2 2 1 1 1 1 1 1 1 1 1 1
     2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1
+---O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O---+
|                                                 |
+---O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O---+
     Z Y X W V U T S R P N M L K J H F E D C B A
```

Porta Audio/Video

Pin	Tipo	Nota
1	LUMINANZA	
2	GND	
3	AUDIO OUT	
4	VIDEO OUT	
5	AUDIO IN	

```
    ++ ++
   / +-+ \
  /       \
 +  3   1  +
 | •     • |
 |         |
 +  •   •  +
  \5  •  4/
   \  2  /
    +---+
```

Porta Registratore

Pin	Tipo
A-1	GND
B-2	+5V
C-3	MOTORE REGISTRATORE
D-4	LETTURA CASSETTA
E-5	SCRITTURA CASSETTA
F-6	SENSORE CASSETTA

```
     1 2 3 4 5 6
+---O-O-O-O-O-O---+
|                 |
+---O-O-O-O-O-O---+
     A B C D E F
```

Porta Seriale I/O

Pin	Tipo
1	SERIALE / SRQIN
2	GND
3	USCITA SERIALE ATN
4	INGRESSO/USCITA SERIALE CLK
5	INGRESSO USCITA DATI SERIALE
6	/RESET

```
    ++ ++
   / +-+ \
  /5     1\
 +  •   •  +
 |    6    |
 |    •    |
 |         !
 +  •   •  +
  \4  •  2/
   \  3  /
    +---+
```

Porta I/O Utente

Pin	Tipo	Nota		Pin	Tipo
1	GND			A	GND
2	+5V	MAX 100mA		B	/FLAG2
3	/RESET			C	PB0
4	CNT1			D	PB1
5	SP1			E	PB2
6	CNT2			F	PB3
7	SP2			H	PB4
8	/PC2			I	PB5
9	USCITA SERIALE ATN			K	PB6
10	9 Vac	MAX 100mA		L	PB7
11	9 Vac	MAX 100mA		M	PA2
12	GND			N	GND

```
                      1 1 1
   1 2 3 4 5 6 7 8 9 0 1 2
+--O-O-O-O-O-O-O-O-O-O-O-O--+
|                           |
+--O-O-O-O-O-O-O-O-O-O-O-O--+
   A B C D E F H J K L M N
```

APPENDICE J

PROGRAMMI DA PROVARE

In questa appendice abbiamo incluso alcuni programmi utili da provare con il tuo Commodore 64.

```
start tok64 jotto.prg
 100 print "{down}jotto{space*7}jim butterfield"
 120 input "{down}want instructions";z$:if asc(z$)=78 goto 250
 130 print "{down}try to guess the mystery 5-letter word"
 140 print "{down}you must guess only legal 5-letter"
 150 print "words, too..."
 160 print "you will be told the number of matches"
 170 print "(or 'jots') of your guess."
 180 print "{down}hint: the trick is to vary slightly"
 190 print "  from one guess to the next; so that"
 200 print "  if you guess 'batch' and get 2 jots"
 210 print "  you might try 'botch' or 'chart'"
 220 print "  for the next guess..."
 250 data bxbsf,ipccz,dbdif,esfbe,pggbm
 260 data hpshf,ibudi,djwjm,kpmmz,lbzbl
 270 data sbkbi,mfwfm,njnjd,boofy,qjqfs
 280 data rvftu,sjwfs,qsftt,puufs,fwfou
 290 data xfbwf,fyupm,nvtiz,afcsb,gjaaz
 300 data uijdl,esvol,gmppe,ujhfs,gblfs
 310 data cppui,mzjoh,trvbu,hbvaf,pxjoh
 320 data uisff,tjhiu,bymft,hsvnq,bsfob
 330 data rvbsu,dsffq,cfmdi,qsftt,tqbsl
 340 data sbebs,svsbm,tnfmm,gspxo,esjgu
 400 n=50
 410 dim n$(n),z(5),y(5)
 420 for j=1 to n: read n$(j): next j
 430 t=ti
 440 t=t/1000:if t>=1 then goto 440
 450 z=rnd(-t)
 500 g=0: n$=n$(rnd(1)*n+1)
 510 print "{down}i have a five letter word:": if r>0 then 560
 520 print "guess (with legal words)"
 530 print "and i'll tell you how many"
 540 print "'jots', or matching letters,"
 550 print "you have...."
 560 g=g+1: input "your word";z$
 570 if len(z$)<>5 then print "you must guess a 5-letter word!": goto 560
 580 v=0: h=0: m=0
 590 for j=1 to 5
 600 z=asc(mid$(z$,j,1)): y=asc(mid$(n$,j,1))-1: if y=64 then y=90
```

```
 610 if z<65 or z>90 then print "that's not a word!": goto 560
 620 if z=65 or z=69 or z=73 or z=79 or z=85 or z=89 then v=v+1
 630 if z=y then m=m+1
 640 z(j)=z: y(j)=y: next j
 650 if m=5 goto 800
 660 if v=0 or v=5 then print "come on..what kind of a word is that?": goto 560
 670 for j=1 to 5: y=y(j)
 680 for k=1 to 5: if y=z(k) then h=h+1:z(k)=0:goto 700
 690 next k
 700 next j
 710 print"{up}{right*19}";h;"jots"
 720 if g<30 goto 560
 730 print "i'd better tell you.. word was '";
 740 for j=1 to 5: print chr$(y(j));: next j
 750 print"'": goto 810
 800 print "you got it in only";g;"guesses."
 810 input "{down}another word";z$
 820 r=1: if asc(z$)<>78 goto 500
stop tok64
```


begin 644 jotto.prg
M`0@F"&0`F2`B$4I/5%1/("`@("`@($I)32!"551415)&245,1"(`50AX`(4@
M(A%704Y4($E.4U1254-424].4R([6B0ZBR#&*%HD*;(W."")(#(U,`"%"((`
MF2`B$51262!43R!'54534R!42$4@35E35$5262`U+4Q%5%1%4B!73U)$(@"Q
M"(P`F2`B$5E/52!-55-4($=515-3($].3%D@3$5'04P@-2U,151415(B`,<(
ME@"9(")73U)$4RP@5$]/+BXN(@#V"*``F2`B64]5(%=)3$P@0D4@5$],1"!4
M2$4@3E5-0D52($]&($U!5$-(15,B`!D)J@"9("(H3U(@)TI/5%,G*2!/1B!9
M3U52($=515-3+B(`1@FT`)D@(A%(24Y4.B!42$4@5%))0TL@25,@5$\@5D%2
M62!33$E'2%1,62(`=`F^``)D@(B`@1E)/32!/3D4@1U5%4U,@5$\@5$A%($Y%
M6%0[(%-/(%1(050B`*()R`"9("(@($E&(%E/52!'54534R`G0D%40T@G($%.
M1"!'150@,B!*3U13(@#-"=(`F2`B("!93U4@34E'2%0@5%)9(`=")3U1#2"<@
M3U(@)T-(05)4(`[0G<`)D@(B`@1D]2(%1(12!.15A4($=515-3+BXN(@`1
M"OH`@R!"6$)31BQ)4$-#6BQ$0D1)1BQ%4T9"12Q01T=")30`U"@0!@R!(4%-(
M1BQ)0E5$22Q$2E=*32Q+4$U-6BQ,0EI"3`!9"@X!@R!30DM"22Q-1E=&32Q.
M2DY*1"Q"3T]]&62Q12E%&4P!]"A@!@R!25D9452Q$32E=&4RQ14T945"Q0555&
M4RQ&5T9/50"A"B(!@R!81D)71BQ&655032Q$0BQ!1D-30BQ'2D%!6@#%
M"BP!@R!524I$3"Q%4U9/3"Q'35!012Q52DA&4RQ'0DQ&4#I"C8!@R!#4%%!5
M22Q-6DI/2"Q44E9"52Q(0E9!1BQ06$9!1BQ42DA)52Q+4$]!1BQ,0E9)1BQ.
M64U&5"Q(4E9/50`Q"TH!@R!25D)352Q(4D]"5"Q34E!.24Y6"Q$22Q14T=,
M5"Q444)).4DL1DY%1"Q'25E&2RQ02D=$3$1;"TY!@R!45E!/4"Q"2%)$5"Q5
M"Y`!3K(U,`!T"YH!AB!.)"A.*2Q=2D)\ D/ND`8$@2K(Q(*0@3CH@
MAR!.)"A**3H@@B!*`)D+K@%4LE1)`+4+N`%4LE2M,3`P,#J+(%2QLC$@IR")
M(#0T,`#!"\(!6K*M4*0@0!1[(P.B!.)+),(P!$"`B[*#IKZJ,2D`"PS^
M`9D@(A%)($A!5D4@02!73U)$(%Q%5%1%4B!73U)$`.B(Z((L@4K$P@0D4K$P@-38P
M`"P,"`*9(")'54534R!,14942"!73U)$4RDB`$\,$@*9(")!3D0@22!7259,
M(22=,3"!414Q,(%E/52!(3U<@34%.62`I3@34%4`Q,`,34U4(%=/4D0@
M0TA)3D$N(`)D,*@`@+@`I(@*9*")93U4@2$%612`U,`53(%-(0D!-2`@
M,3H@A2`B64]5(%=/4D0B.R!-(`"3##X`F2`B$64]5(%=/4D0@5T%3(%$@3E1A
M55-4($=515-3("=!-4@T2U@!!P$(@(!-U$@.SE9(E1%3$@%9$@9$@@9
M-0"U#$("H2!:)"#\`@Q!#T.6Z`PXP$82(*\`I!]4SUA(#T`I!`A9%<&5&4V0@
M.C!6)3(@1!D@1$1E-2X!2!`N2!3,&`R!&3`=`P+2Z@PA`X!$2@P4*&CB$
M,%9,&D`X"(*1`%`(*IE$22`@4RMN:G`CH(`%@`2#SA(@(P!5$R4`2`@($D`
MU2Z8@#(IO,N%*`Q"`Z3`5*(P`%BXN-D<@K9`0@B;X!D5`B&02@RK2`H@`A
MA2`?4D4H(&90@0"*!4@2`&8J#OAMRE(@K2`D22`K-5`E2BD@T-*R2@@,2E
M%26(@1$Z4@`3@FT*(P`=Z*5@#&F`J%F@8*)@E@8&)E8,1AH@``

M1T]4($E4($E.($].3%DB.T<[(D=515-315,N(@`]#RH#A2`B$4%.3U1(15(@
F5T]21"([6B0`6`\T`U*R,3H@BR#&*%HD*;.Q-S@@B2`U,#``````
`

end

```
start tok64 sequence.prg
 1 rem *** sequence
 2 rem
 3 rem *** from pet user group
 4 rem *** software exchange
 5 rem *** po box 371
 6 rem *** montgomeryville, pa 18936
 7 rem ***
 50 dim a$(26)
 100 z$="abcdefghijklmnopqrstuvwxyz"
 110 z1$="12345678901234567890123456"
 200 print "{clear}{down*2}inserisci la lunghezza della stringa da
            sequenziare{down}"
 220 input "la lunghezza massima è 26 ";s%
 230 if s%<1 or s%>26 then 200
 240 s=s%
 300 for i=1 to s
 310 a$(i)=mid$(z$,i,1)
 320 next i
 400 rem randomize string
 420 for i=1 to s
 430 k=int(rnd(1)*s+1)
 440 t$=a$(i)
 450 a$(i)=a$(k)
 460 a$(k)=t$
 470 next i
 480 gosub 950
 595 t=0
 600 rem rverse substring
 605 t=t+1
 610 input "quanti da invertire ";r%
 620 if r%=0 goto 900
 630 if r%>0 and r%<=s goto 650
 640 print "deve essere tra 1 e ";s: goto 610
 650 r=int(r%/2)
 660 for i=1 to r
 670 t$=a$(i)
 680 a$(i)=a$(r%-i+1)
 690 a$(r%-i+1)=t$
 700 next i
 750 gosub 950
 800 c=1: for i=2 to s
 810 if a$(i)>a$(i+1) goto 830
 820 c=0
 830 next i
 840 if c=0 goto 600
```

```
 850 print "{down}l'hai fatto in ";t;" tentativi"
 900 rem check for another game

 910 input "{down}vuoi riprovare ";y$
 920 if left$(y$,1)="y" or y$="ok" or y$="1" goto 200
 930 end
 950 print
 960 print left$(z1$,s)
 970 for i=1 to s: print a$(i);: next i
 980 print "{down}"
 990 return
stop tok64
```


```
begin 644 sequence.prg
M`0@4"`$`CR`J*BH@4T51545.0T4`&@@"`(\`.`@#`(\@*BHJ($923TT@4$54
M(%5315(@1U)/55``5`@$`(\@*BHJ(%-/1E1705)%($580TA!3D=%`&D(!0"/
M("HJ*B!03R!"3U@@,S<Q`(T(!@"/("HJ*B!-3TY41T]-15)95DE,3$4L(%!!
M(#$X.3,,V`)<(!P"/("HJ*@"D"#(`AB!!)"@R-BD`R`AD`%HDLB)!0D-$149'
M2$E*2TQ-3D]045)35%565UA96B(`[0AN`%HQ)+(B,3(S-#4V-S@Y,#$R,S0U
M-C<X.3`Q,C,T-38B`"`)R`"9("*3$1%%3E1%4B!,14Y'5$@@3T8@4U1224Y'
M(%1/($)%(%(%-%455%3D-%1!$B`$$)W`"%(")-05A)355-($Q%3D=42"!)4R`R
M-B`B.U,E`%H)Y@"+(%,ELS$@L"!3);$R-B"G(#(P,`!C"?``4[)3)0!Q"2P!
M@2!)LC$@I"!3`(4)-@%!)"A)*;+**%HD+$DL,2D`C0E``8(@20"D"9`!CR!2
M04Y$3TU)6D4@4U1224Y'`+()I`&!($FR,2"D(%,`Q`FN`4NRM2B[*#$IK%.J
M,,2D`T0FX`50DLD$D*$DI`.$)P@%!)"A)*;)!)"A+*0#N"<P!020H2RFR5"0`
M]@G6`8(@20``"N`!C2`Y-3``"`I3`E2R,``?"E@"CR!25D524T4@4U5"4U12
M24Y'`"D*70)4LE2J,0!)"F("A2`B2$]7(%U!3ED@5$\@4D5615)312`B.U(E
M`%H*;`*+(%(ELC`@B2`Y,#``<PIV`HL@4B6Q,,""O(%(EL[)3((D@-C4P`)L*
M@`*9(")-55-4($)%(%5%5=%14X@,2!!3D0@(CM3.B")(#8Q,`"I"HH"4K*U
M*%(EK3(I`+<*E`*!($FR,2"D(%(`J`Q`J>`E0DLD$D*$DI`-D*J`)!)"A)*;)!
M)"A2):M)JC$I`.L*L@)!)"A2):M)JC$ILE0D`/,*O`*"(%D`_0KN`HT@.34P
M`!`+(`-#LC$Z(($@2;(*0@4P`J"RH#BR!!)"A)*;%!)"A)JC$I`&AJC$I((D@.#,P
M`#(+(+-`-#LC``.@L^``X(@20!*"T@#BR!#LC`@B2`V,#``;0M2`YD@(A%93U4@
M1$E$($E4($E.("([5#LB(%122453(@"*"X0#CR!#2$5#2R!&3U(@04Y/5$A%
M4B!'04U%`*H+C@.%("(15T%.8(%1262!!43$R$%'04E.("([620`U0N8`XL@
MR"A9)"PQ*;(B62(@L"!9)+(B3TLB(+`@62*RB(Q(B(*B@(#(P,`#;"Z(#@`#A
MM@.9`/`+I@.9(,@H6C$D+%,I``T,J`.!($FR,2"D(%,Z()D@020H22D[.B""
5($D`%PS4`YD@(A$B`!T,W@..````
`
end
```


start tok64 pianokey.prg

```
 90 REM piano keyboard
 100 PRINT"{CLEAR} {REVERSE ON} {RIGHT} {RIGHT} {194} {RIGHT} {RIGHT} \
      {RIGHT} {194} {RIGHT} {RIGHT} {194} {RIGHT} {RIGHT} "
 110 PRINT" {REVERSE ON} {RIGHT} {RIGHT} {194} {RIGHT} {RIGHT} {RIGHT} \
      {194} {RIGHT} {RIGHT} {194} {RIGHT} {RIGHT} "
 120 PRINT" {REVERSE ON} {RIGHT} {RIGHT} {194} {RIGHT} {RIGHT} {RIGHT} \
      {194} {RIGHT} {RIGHT} {194} {RIGHT} {RIGHT} "
 130 PRINT" {REVERSE ON} {194} {194} {194} {194} {194} {194} {194} \
     {194} {194} {194} {194} {194} "
 140 PRINT" {REVERSE ON}q{194}w{194}e{194}r{194}t{194}y{194}u{194} \
      i{194}o{194}p{194}@{194}*{194}^"
 150 PRINT"{DOWN}'space' for solo or polyphonic"
 160 PRINT"{DOWN}'f1,f3,f5,f7' octave selection"
 170 PRINT"{DOWN}'f2,f4,f6,f8' waveform{DOWN}"
 180 PRINT"hang on, setting up frequency table..."
 190 S=13*4096+1024: DIM F(26): DIM K(255)
```

```
 200 FOR I=0 TO 28: POKE S+I,0: NEXT
 210 F1=7040: FOR I=1 TO 26: F(27-I)=F1*5.8+30: F1=F1/2^(1/12): NEXT
 220 K$="q2w3er5t6y7ui9o0p@-*\^"
 230 FOR I=1 TO LEN(K$): K(ASC(MID$(K$,I)))=I: NEXT
 240 PRINT "{UP}{SPACE*38}"
 250 AT=0:DE=0:SU=15:RE=9:SV=SU*16+RE:AV=AT*16+DE:WV=16:W=0:M=1:OC=4:\
     HB=256:Z=0
 260 FOR I=0 TO 2: POKE S+5+I*7,AT*16+DE: POKES+6+I*7,SU*16+RE
 270 POKE S+2+I*7,4000 AND 255: POKE S+3+I*7,4000/256: NEXT
 280 POKE S+24,15: REM+16+64:poke s+23,7
 300 GET A$:IF A$="" THEN 300
 310 FR=F(K(ASC(A$)))/M: T=V*7: CR=S+T+4: IF FR=Z THEN 500
 320 POKE S+6+T,Z: REM finish dec/sus
 325 POKE S+5+T,Z: REM finish att/rel
 330 POKE CR,8: POKE CR,0: REM fix off
 340 POKE S+T,FR-HB*INT(FR/HB): REM set lo
 350 POKE S+1+T,FR/HB: REM set hi
 360 POKE S+6+T,SV: REM set dec/sus
 365 POKE S+5+T,AV: REM set att/rel
 370 POKE CR,WV+1: FOR I=1 TO 50*AT: NEXT
 375 POKE CR,WV: REM pulse
 380 IF P=1 THEN V=V+1: IF V=3 THEN V=0
 400 GOTO 300
 500 IF A$="{F1}" THEN M=1: OC=4: GOTO 300
 510 IF A$="{F3}" THEN M=2: OC=3: GOTO 300
 520 IF A$="{F5}" THEN M=4: OC=2: GOTO 300
 530 IF A$="{F7}" THEN M=8: OC=1: GOTO 300
 540 IF A$="{F2}" THEN W=0: WV=16: GOTO 300
 550 IF A$="{F4}" THEN W=1: WV=32: GOTO 300
 560 IF A$="{F6}" THEN W=2: WV=64: GOTO300
 570 IF A$="{F8}" THEN W=3: WV=128: GOTO300
 580 IF A$=" " THEN P=1-P: GOTO 300
 590 IF A$="{CLEAR}" THEN 200
 600 GOTO 300
 800 PRINT"hit a key"
 810 GET A$: IF A$="" THEN 810: WAIT FOR A KEY
 820 PRINT A$: RETURN
stop tok64
```


```
begin 644 pianokey.prg
M`0@6"%H`CR!024%.3R!+15E"3T%21``Z"&0`F2*3(!(@'2`=(,(@'2`=(!T@
MPB`=(!T@PB`=(!T@(@!="&X`F2(@$B`=(!T@PB`=(!T@'2#"(!T@'2#"(!T@
M'2`B`(`(>`"9(B`2(!T@'2#"(!T@'2`=(,(@'2`=(,(@'2`=("(`HPB"`)DB
M(!(@PB#"(,(@PB#"(,(@PB#"(,(@(@#&"(P`F2(@$E'"5\)%PE+"
M5,)9PE7"2<)/PE#"0,(JPEXB`.T(E@"9(A$G4U!!0T4G($9/4B!33TQ/($]2
M(%!/3%E02$]./24,B`!0)H`"9(A$G1C$L1C,L1C4L1C<G($]#5$%612!314Q%
M0U1)3TXB`#0)J@"9(A$G1C(L1C0L1C8L1C@G(%=!5D5&3U)-$2(`8@FT`)DB
M2$%.1R!/3BP@4T545$E.1R!54"!&4D545.0UD@5$%"3$4N+BXB`(@)O@!3
MLC$SK#0P.3:J,3`R-#H@AB!&*#(V*3H@AB!+*#(U-2D`HPG(`($@2;(P(*0@
M,C@Z()<@4ZI)+#`Z(((`X0G2`$8QLC<P-#`Z(($@2;(Q(*0@,C8Z($8H,C>K
M22FR1C&L-2XXJC,P.B!&,;)&,:TRKB@QK3$R*3H@@@`!"MP`2R2R(E$R5S-%
M4C54-EDW54DY3S!00"TJ7%XB`"<*Y@"!($FR,2"D(,,H2R0I.B!+*,8HRBA+
M)"Q)*2DILDDZ((`5PKP`)D@(I$@("`@("`@("`@("`@("`@("`@("`@("`@
M("`@("`@("`@("(`I@KZ`$%4LC`Z1$6R,#I35;(Q-3I21;(Y.E-6LE-5
MK#$VJE)%.D6LD%4K#$VJD1%.E=6LC$V.E>R,#I-LC$Z3T.R-#I(0K(R-38Z
M6K(P`-L*!`&!($FR,""D(#(Z()<@4ZHUJDFL-RQ!5*PQ-JI$13H@EU.J-JI)
MK#<L4U6L,3:J4D4`"PL.`9<@4ZHRJDFL-RPT,#`P(*\@,C4U.B"7(%.J,ZI)
MK#<L-#`P,*TR-38Z((`+@L8`9<@4ZHR-"PQ-3H@CRLQ-BLV-#I03TM%(%,K
M,C,L-P!%"RP!H2!!)#J+($$DLB(B(*<@,S`P`'D+-@%&4K4K)&*$LHQBA!)"DI
M*:U-.B!4LE:L-SH@0U*R4ZI4JC0Z((L@1E*R6B"G(#4P,`"9"T`!ER!3JC:J
M5"Q:.B"/($9)3DE32"!$14,O4U53`+D+10&7(%.J-J:I4+%HZ((\@1DE.25-(
M($%45"]214P`UPM*`9<@0U(L.#H@ER!#4BPP.B"/($9)6"!/1D8`^@M4`9<@
M4ZI4+$92JTA"M2Y5JT92K4A"*3H@CR!3150@3$\`%@Q>`9<@4ZHQJE0L1E*M
M2$(Z((\@4T54($A)`#H,:`&7(%.J-JI4+%-6.B"/($1%0R]355,`7@QR`9<@
```

```
M0CH@CR!3150@2$D`-`QH`9<@4ZHVJE0L4U8Z((\@4T54($1%0R]355,`4@QM
M`9<@4ZHUJE0L058Z((\@4T54($%45"]214P`<@QR`9<@0U(L5U:J,3H@@2!)
MLC$@I"`U,*Q!5#H@@@"'#'<!ER!#4BQ75CH@CR!054Q310"F#'P!BR!0LC$@
MIR!6LE:J,3H@BR!6LC,@IR!6LC``L`R0`8D@,S`P`-`,]`&+($$DLB*%(B"G
M($VR,3H@3T.R-#H@B2`S,#``\`S^`8L@022R(H8B(*<@3;(R.B!/0[(S.B")
M(#,P,``0#0@"BR!!)+(BAR(@IR!-LC0Z($]#LC(Z((D@,S`P`#`-$@*+($$D
MLB*((B"G($VR.#H@3T.R,3H@B2`S,#``40T<`HL@022R(HDB(*<@5[(P.B!7
M5K(Q-CH@B2`S,#``<@TF`HL@022R(HHB(*<@5[(Q.B!75K(S,CH@B2`S,#``
MD@TP`HL@022R(HLB(*<@5[(R.B!75K(V-#H@B3,P,`"S#3H"BR!!)+(BC"(@
MIR!7LC,Z(%=6LC$R.#H@B3,P,`#/#40"BR!!)+(B("(@IR!0LC&K4#H@B2`S
M,#``X@U.`HL@022R(I,B(*<@,C`P`.P-6`*)(#,P,`#]#2`#F2)(250@02!+
M15DB`"`.*@.A($$D.B"+($$DLB(B(*<@.#$P.B"2(($@02!+15D`+`XT`YD@
(020Z((X`````
`
end
```


APPENDICE K

CONVERSIONE DI PROGRAMMI BASIC STANDARD IN BASIC COMMODORE 64

Se hai programmi scritti in un BASIC diverso da quello Commodore, prima di eseguirli sul Commodore possono essere necessari alcuni adattamenti. Per rendere la conversione più facile abbiamo incluso alcuni suggerimenti.

Dimensionamenti delle Stringhe

Cancellare tutte le istruzioni che vengono utilizzate per dichiarare la lunghezza delle stringhe. Una istruzione come DIM A$(I,J), che dimensiona un vettore di stringa di J elementi di lunghezza I, dovrà essere convertito nel BASIC Commodore come istruzione DIM A$(J).
Alcuni BASIC utilizzano una virgola o una & per la concatenazione di stringhe. Ognuno di questi operatori deve essere sostituito con un segno +, che è l'operatore del BASIC Commodore per la concatenazione di stringhe.
Nel BASIC Commodore 64 le funzioni MID$, RIGHT$ e LEFT$ si utilizzano per creare sotto-stringhe di stringhe. Le forme come A$(I) per accedere al I esimo carattere in A$, o A$(I,J) per creare una sottostringa di A$ dalla posizione I alla J, devono essere sostituite come segue:

Altri BASIC	BASIC Commodore 64
A$(I)=X$	A$=LEFT$(A$,I-1)+X$+MID$(A$,I+1)
A$(I,J)=X$	A$=LEFT$(A$,I-1)+X$+MID$(A$,J+1)

Assegnazioni Multiple

Per impostare B e C uguale a zero, alcuni BASIC permettono istruzioni in questa forma:

```
10 LET B=C=0
```

Il BASIC del Commodore 64 interpreterebbe il secondo segno dell'uguale come un operatore logico ed imposterà B = -1 se C = 0. Al posto di questa istruzione convertire in:

```
10 C=0 : B=0
```

Istruzioni Multiple

Alcuni BASIC utilizzano una barra inversa per separare più istruzioni su un riga. Con il BASIC del Commodore 64 separare tutte le istruzioni con i due punti (:).

Funzioni MAT

I programmi che usano le funzioni MAT, disponibili su alcuni BASIC, affinché funzionino correttamente devono essere riscritti utilizzando i cicli FOR...NEXT.

APPENDICE L

MESSAGGI D'ERRORE

Questa appendice comprende un elenco completo dei messaggi d'errore generati dal Commodore 64, con la descrizione della causa.

BAD DATA	Da un file aperto sono stati ricevuti dati di stringa, ma il programma aspettava dati numerici.
BAD SUBSCRIPT	Il programma tentava di far riferimento ad un elemento di una matrice il cui numero è fuori dal campo indicato dall'istruzione DIM.
BREAK	L'esecuzione del programma è stata arrestata perché hai premuto il tasto <STOP>.
CAN'T CONTINUE	Il comando CONT non funzionerà perché il programma non è stato eseguito (RUN) - c'è stato un errore - oppure è stata editata una riga.
DEVICE NOT PRESENT	Il dispositivo di I/O richiesto non era disponibile per un OPEN, CLOSE, CMD, PRINT#, INPUT# o GET#.
DIVISION BY ZERO	La divisione per zero è una stranezza matematica e non viene permessa.
EXTRA IGNORED	Alla risposta in una istruzione INPUT di dati, sono stati immessi troppi elementi. Vengono accettati solo i primi pochi elementi.
FILE NOT FOUND	Se cercavi un file su nastro, è stato trovato il marcatore END-OF-TAPE. Se cercavi su disco, con quel nome non esiste alcun file.
FILE NOT OPEN	Il file indicato in un CLOSE, CMD, PRINT#, INPUT# o GET#, deve prima essere aperto (OPEN).
FILE OPEN	È stato fatto un tentativo di aprire un file utilizzando il numero di un file già aperto.
FORMULA TOO COMPLEX	Affinché il sistema possa operare, l'espressione di stringa da valutare dovrà essere suddivisa in almeno due parti, oppure una formula ha troppe parentesi.
ILLEGAL DIRECT	L'istruzione INPUT può essere utilizzata solo all'interno di un programma e non in modalità immediata.
ILLEGAL QUANTITY	Un numero usato come argomento di una funzione o di una istruzione è fuori dal campo ammissibile.
LOAD	C'è un problema con il programma su nastro.
NEXT WITHOUT FOR	Questo errore è causato o da una nidificazione scorretta di cicli, o per avere il nome della variabile, in una istruzione NEXT, che non corrisponde a quello dell'istruzione FOR.
NOT INPUT FILE	È stato fatto un tentativo di prelevare (INPUT o GET) dati da un file che era indicato solo d'uscita.

NOT OUTPUT FILE	È stato fatto un tentativo di stampare (PRINT) dati su un file che era indicato solo come d'ingresso.
OUT OF DATA	È stata eseguita una istruzione READ ma nell'istruzione DATA non sono rimasti dati da leggere.
OUT OF MEMORY	Non c'è più RAM disponibile per il programma o le variabili. Questo errore può capitare anche quando siano stati nidificati troppi cicli FOR, oppure troppi GOSUB.
OVERFLOW	Il risultato di un calcolo è maggiore del numero più grande permesso, che è 1.70141884E+38.
REDIM'D ARRAY	Una matrice può essere dimensionata solo una volta. Se viene utilizzata una variabile di matrice prima che venga DIMensionata detta matrice, su quella matrice verrà eseguita una operazione automatica di DIMensionamento che imposta a dieci la quantità degli elementi; qualsiasi successivo DIMensionamento provocherà questo errore.
REDO FROM START	Durante una istruzione INPUT sono stati immessi dati carattere quando il programma aspettava dati numerici. Basterà ridigitare correttamente i dati in ingresso ed il programma continuerà da solo.
RETURN WITHOUT GOSUB	È stata incontrata una istruzione RETURN senza aver prima trovato nessun comando GOSUB.
STRING TOO LONG	Una stringa può contenere fino a 255 caratteri.
SYNTAX ERROR	Il Commodore 64 non riconosce una istruzione. La causa può essere una mancanza o un eccesso di parentesi, un errore di ortografia su una parola-chiave, ecc.
TYPE MISMATCH	Questo errore capita quando viene utilizzato un numero al posto di una stringa, o viceversa.
UNDEF'D FUNCTION	Un utente ha definito una funzione fornita di richiami ma non è mai stata definita con l'istruzione DEF FN.
UNDEF'D STATEMENT	È stato fatto un tentativo, con GOTO o GOSUB o RUN, su un numero di riga inesistente.
VERIFY	Il programma su nastro o disco non è uguale al programma attualmente in memoria.

APPENDICE M

VALORI DELLE NOTE MUSICALI

Questa appendice comprende un elenco completo di Note#, nota reale, ed i valori da inserire (POKE) nei registri HI FREQ e LOW FREQ del microcircuito audio per riprodurre la nota indicata.

Nota musicale		Frequenza oscillatore			Nota musicale		Frequenza oscillatore			Nota musicale		Frequenza oscillatore		
Nota	**Ottava**	**Hz**	**Hi**	**Low**	**Nota**	**Ottava**	**Hz**	**Hi**	**Low**	**Nota**	**Ottava**	**Hz**	**Hi**	**Low**
0	DO-0	268	1	12	32	DO-2	1072	4	48	64	DO-4	4291	16	195
1	DO#-0	284	1	28	33	DO#-2	1136	4	112	65	DO#-4	4547	17	195
2	RE-0	301	1	45	34	RE-2	1204	4	180	66	RE-4	4817	18	209
3	RE#-0	318	1	62	35	RE#-2	1275	4	251	67	RE#-4	5103	19	239
4	MI-0	337	1	81	36	MI-2	1351	5	71	68	MI-4	5407	21	31
5	FA-0	358	1	102	37	FA-2	1432	5	152	69	FA-4	5728	22	96
6	FA#-0	379	1	123	38	FA#-2	1517	5	237	70	FA#-4	6069	23	181
7	SOL-0	401	1	145	39	SOL-2	1607	6	71	71	SOL-4	6430	25	30
8	SOL#-0	425	1	169	40	SOL#-2	1703	6	167	72	SOL#-4	6812	26	156

9	LA-0	451	1	195	41	LA-2	1804	7	12	73	LA-4	7217	28	49
10	LA#-0	477	1	221	42	LA#-2	1911	7	119	74	LA#-4	7647	29	223
11	SI-0	506	1	250	43	SI-2	2025	7	233	75	SI-4	8101	31	165
16	DO-1	536	2	24	48	DO-3	2145	8	97	80	DO-5	8583	33	135
17	DO#-1	568	2	56	49	DO#-3	2273	8	225	81	DO#-5	9094	35	134
18	RE-1	602	2	90	50	RE-3	2408	9	104	82	RE-5	9634	37	162
19	RE#-1	637	2	125	51	RE#-3	2551	9	247	83	RE#-5	10207	39	223
20	MI-1	675	2	163	52	MI-3	2703	10	143	84	MI-15	10814	42	62
21	FA-1	716	2	204	53	FA-3	2864	11	48	85	FA-5	11457	44	193
22	FA#-1	758	2	246	54	FA#-3	3034	11	218	86	FA#-5	12139	47	107
23	SOL-1	803	3	35	55	SOL-3	3215	12	143	87	SOL-5	12860	50	60
24	SOL#-1	851	3	83	56	SOL#-3	3406	12	78	88	SOL#-5	13625	53	57
25	LA-1	902	3	134	57	LA-13	3608	14	24	89	LA-5	14435	56	99
26	LA#-1	955	3	187	58	LA#-3	3823	14	239	90	LA#-5	15294	59	190
27	SI-1	1012	3	244	59	SI-3	4050	15	210	91	SI-5	16203	63	75

Nota musicale		Frequenza oscillatore		
Nota	**Ottava**	**Hz**	**Hi**	**Low**
96	DO-6	17167	67	15
97	DO#-6	18188	71	12
98	RE-6	19269	75	69
99	RE#-6	20415	79	191
100	MI-6	21629	84	125
101	FA-6	22915	89	131
102	FA#-6	24278	94	214
103	SOL-6	25721	100	121
104	SOL#-6	27251	106	115
105	LA-6	28871	112	199
106	LA#-6	30588	119	124
107	SI-6	32407	126	151
112	DO-7	34334	134	30
113	DO#-7	36376	142	24
114	RE-7	38539	150	139
115	RE#-7	40830	159	126
116	MI-7	43258	168	250
117	FA-7	45830	179	6
118	FA#-7	48556	189	172
119	SOL-7	51443	200	243
120	SOL#-7	54502	212	230
121	LA-7	57743	225	143
122	LA#-7	61176	238	248
123	SI-7	64874	253	46

Impostazione Filtri

Locazione	Contenuti
54293	Taglio frequenza bassa (0-7)
54294	Taglio frequenza alta (0-255)
54295	Risonanza (bit da 4 a 7)
	Filtro voce 3 (bit 2)
	Filtro voce 2 (bit 1)
	Filtro voce 1 (bit 0)
54296	Passa-alto (bit6)
	Passa-banda (bit 5)
	Passa-basso (bit 4)
	Volume (bit da 0 a 3)

APPENDICE N

BIBLIOGRAFIA

Addison-Wesley	"BASIC and the Personal Computer", Dwyer e Critchfield
Compute	"Compute's First Book of PET/CBM"
Cowbay Computing	"Feed Me, I'm Your PET Computer", Carol Alexander
	"Looking Good with Your PET", Carol Alexander
	"Teacher's PET -- Plans, Quizzes, and Answers"
Creative Computing	"Getting Acquainted With Your VIC 20", T. Hartnell
Dilithium Press	"BASIC Basic-English Dictionary for the PET", Lorry Noonan
	"PET BASIC", Tom Rugg and Phil Feldman
Faulk Baker Associates	"MOS Programming Manual", MOS Technology
Hoyden Book Co.	"BASIC From the Ground Up", David E. Simon
	"I Speak BASIC to My PET", Aubrey Jones, Jr.
	"Library of PET Subroutines',', Nick Hampshire
	"PET Graphics", Nick Hampshire
	"BASIC Conversions Handbook, Apple, TRS-80, and PET", David A. Brain, Phillip R. Oviatt, Paul J. Paquin e Chandler P. Stone
Howard W. Sams	"The Howard W. Sams Crash Course in Microcomputers", Louis E. Frenzel, Jr.
	"Mostly BASIC: Applications for Your PET", Howard Berenbon
	"PET Interfacing", James M. Downey e Steven M. Rogers
	"VIC 20 Programmer's Reference Guide", A. Finkel, P. Higginbottom, N. Harris e M. Tomczyk

Level Ltd.	"Programming the PET/CBM", Raeto West.
Little, Brown & Co.	"Computer Games for Businesses, Schools, and Homes", J. Victor Nagigian e William S. Hodges
	"The Computer Tutor: Learning Activities for Homes and Schools", Gary W. Orwig, University of Central Florida e William S. Hodges
McGraw-Hill	"Hands-On BASIC With a PET", Herbert D. Peckman
	"Home and Office Use of VisiCalc", D. Castlewitz e L. Chisauki
Osborne/McGraw-Hill	"PET/CBM Personal Computer Guide", Carroll S. Donahue
	"PET Fun and Games", R. Jeffries e G. Fisher
	"PET and the IEEE", A. Osborne e C. Donahue
	"Some Common BASIC Programs for the PET", L. Poole, M. Borchers e C. Donahue
	"Osborne CP/M User Guide", Thorn Hogan
	"CBM Professional Computer Guide"
	"The PET Personal Guide"
	"The 8086 Book", Russell Rector and George Alexy
P. C. Publications	"Beginning Self-Teaching Computer Lessons"
Prentice-Hall	"The PET Personal Computer for Beginners", S. Dunn e V. Morgan
Reston Publishing Co.	"PET and the IEEE 488 Bus (GPIB)", Eugene Fisher e C. W. Jensen
	"PET BASIC -- Training Your PET Computer", Roman Zamora, Wm. F. Carrie e B. Allbrecht
	"PET Games and Recreation", M. Ogelsby, L. Lindsey e D. Kunkin
	"PET BASIC", Richard Huskell
	"VIC Games and Recreation"
Telmas Courseware	"BASIC and the Personal Computer", T. A. Dwyer, Ratings e M. Critchfield
Total Information	"Understanding Your PET/CBM, Vol. 1, BASIC Services Programming"
	"Understanding Your VIC", David Schultz

I periodici del Commodore ti forniscono le informazioni più aggiornate per il tuo Commodore 64. Due delle pubblicazioni più popolari alle quali dovresti considerare seriamente di abbonarti sono:

COMMODORE -- Il Periodico per Microcomputer viene pubblicato ogni due mesi ed è disponibile con abbonamento ($15.00 per un anno in U.S., e $25.00 per un anno nel resto del mondo).

POWER/PLAY -- Il Periodico dell'Home Computer viene pubblicato trimestralmente ed è disponibile con abbonamento ($10.00 per un anno in U.S., e $15.00 per un anno nel resto del mondo).

APPENDICE O

MAPPA DEI REGISTRI DEGLI SPRITE

Registro# Dec	Hex	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	S1X7							S0X0	Componente X sprite 0
1	1	S0X7							S0Y0	Componente Y sprite 0
2	2	S1x7							S1X0	Componente X sprite 1
3	3	S1y7							S1Y0	Componente Y sprite 1
4	4	S2X7							S2X0	Componente X sprite 2
5	5	S2Y7							S2Y0	Componente Y sprite 2
6	6	S3X7							S3X0	Componente X sprite 3
7	7	S3Y7							S3Y0	Componente Y sprite 3
8	8	S4X7							S4X0	Componente X sprite 4
9	9	S4Y7							S4Y0	Componente Y sprite 4
10	A	S5X7							S5X0	Componente X sprite 5
11	B	S5Y7							S5Y0	Componente Y sprite 5
12	C	S6X7							S6X0	Componente X sprite 6
13	D	S6Y7							S6Y0	Componente Y sprite 6
14	E	S7X7							S7X0	Componente X sprite 7
15	F	S7Y7							S7Y0	Componente Y sprite 2
16	10	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	Coordinata MSB di X
17	11	RC8	ECM	BMM	BLNK	RSEL	YSCL2	YSCL1	YSCL0	Modalità scorrimento Y
18	12	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	Raster
19	13	LPX7							LPX0	Coord. X penna ottica
20	14	LPY7							LPY0	Coord. Y penna ottica
21	15	SE7							SE0	Abilita sprite (on/off)
22	16	n.c.	n.c.	RST	MCM	CSEL	XSCL2	XSCL1	XSCL0	Modalità scorrimento X
23	17	SEX7							SEX0	Espansione Y sprite
24	18	VS13	VS12	VS11	VS10	CB13	CB12	CB11	n.c.	Memoria carattere di schermo
25	19	IRQ	n.c.	n.c.	n.c.	LPIRQ	ISSC	ISBC	RIIRQ	Richiesta di interrupt
26	1A	n.c.	n.c.	n.c.	n.c.	MLPI	MISSC	MISBC	MRIRQ	Maschera richiesta di interrupt

27 1B	BSP7							BSP0	Priorità fondo-sprite
28 1C	SCM7							SCM0	Sel. Sprite multicolore
29 1D	SEXX7							SEXX0	Espansione X sprite
30 1E	SSC7							SSC0	Collisione sprite-sprite
31 1F	SBC7							SBC0	Collisione sprite-fondo

Registro# Dec Hex	**Colore**	**Registro# Dec Hex**	**Colore**
32 20	Colore del bordo	39 27	Colore sprite 0
33 21	Colore 0 del fondo	40 28	Colore sprite 1
34 22	Colore 1 del fondo	41 29	Colore sprite 2
35 23	Colore 2 del fondo	42 2A	Colore sprite 3
36 24	Colore 3 del fondo	43 2B	Colore sprite 4
37 25	Multicolore 0 dello sprite	44 2C	Colorer sprite 5
38 26	Multicolore 1 dello sprite	45 2D	Colore sprite 6
		46 2E	Colore sprite 7

Codice dei colori

Dec	**Hex**	**Colore**	**Dec**	**Hex**	**Colore**
0	0	NERO	8	8	ARANCIO
1	1	BIANCO	9	9	MARRONE
2	2	ROSSO	10	A	ROSSO CHIARO
3	3	CIANO	11	B	GRIGIO 1
4	4	VIOLA	12	C	GRIGIO 2
5	5	VERDE	13	D	VERDE CHIARO
6	6	BLU	14	E	BLU CHIARO
7	7	GIALLO	15	F	GRIGIO 3

LEGENDA:
NELLA MODALITÀ A CARATTERI MULTICOLORI SI PUÒ UTILIZZARE SOLO I COLORI DA 0 A 7.

APPENDICE P

MAPPA DEI REGISTRI DEL MICROCITCUITO 6566/6567 (VIC-II)

I 6566/6567 sono dispositivi di controllo video del colore da utilizza tanto su terminali video per computer quanto su applicazioni per video-giochi. Entrambi i dispositivi comprendono 47 registri di controllo che sono accessibili attraverso una linea di dati standard ad 8-bit del microprocessore (65XX) ed accedono fino a 16K di memoria per la visualizzazione delle informazioni. Qui sono descritte le varie modalità operative e le opzioni di ciascuna modalità.

MAPPA DEI REGISTRI

INDIRIZZO	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIZIONE
00 ($00)	M0X7	M0X6	M0X5	M0X4	M0X3	M0X2	M0X1	M0X0	Posizione X di MOB 0
01 ($01)	M0Y7	M0Y6	M0Y5	M0Y4	M0Y3	M0Y2	M0Y1	M0Y0	Posizione Y di MOB 0
02 ($02)	M1X7	M1X6	M1X5	M1X4	M1X3	M1X2	M1X1	M1X0	Posizione X di MOB 1
03 ($03)	M1Y7	M1Y6	M1Y5	M1Y4	M1Y3	M1Y2	M1Y1	M1Y0	Posizione Y di MOB 1
04 ($04)	M2X7	M2X6	M2X5	M2X4	M2X3	M2X2	M2X1	M2X0	Posizione X di MOB 2
05 ($05)	M2Y7	M2Y6	M2Y5	M2Y4	M2Y3	M2Y2	M2Y1	M2Y0	Posizione Y di MOB 2
06 ($06)	M3X7	M3X6	M3X5	M3X4	M3X3	M3X2	M3X1	M3X0	Posizione X di MOB 3
07 ($07)	M3Y7	M3Y6	M3Y5	M3Y4	M3Y3	M3Y2	M3Y1	M3Y0	Posizione Y di MOB 3
08 ($08)	M4X7	M4X6	M4X5	M4X4	M4X3	M4X2	M4X1	M4X0	Posizione X di MOB 4
09 ($09)	M4Y7	M4Y6	M4Y5	M4Y4	M4Y3	M4Y2	M4Y1	M4Y0	Posizione Y di MOB 4
10 ($0A)	M5X7	M5X6	M5X5	M5X4	M5X3	M5X2	M5X1	M5X0	Posizione X di MOB 5
11 ($0B)	M5Y7	M5Y6	M5Y5	M5Y4	M5Y3	M5Y2	M5Y1	M5Y0	Posizione Y di MOB 5
12 ($0C)	M6X7	M6X6	M6X5	M6X4	M6X3	M6X2	M6X1	M6X0	Posizione X di MOB 6
13 ($0D)	M6Y7	M6Y6	M6Y5	M6Y4	M6Y3	M6Y2	M6Y1	M6Y0	Posizione Y di MOB 6
14 ($0E)	M7X7	M7X6	M7X5	M7X4	M7X3	M7X2	M7X1	M7X0	Posizione X di MOB 7
15 ($0F)	M7Y7	M7Y6	M7Y5	M7Y4	M7Y3	M7Y2	M7Y1	M7Y0	Posizione Y di MOB 7
16 ($10)	M7X8	M6X8	M5X8	M4X8	M3X8	M2X8	M1X8	M0X8	MSB della posizione X
17 ($11)	RC8	ECM	BMM	DEN	RSEL	Y2	Y1	Y0	Vedi testo
18 ($12)	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	Registro del raster

INDIRIZZO	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIZIONE
19 ($13)	LPX8	LPX7	LPX6	LPX5	LPX4	LPX3	LPX2	LPX1	Coord. X penna ottica
20 ($14)	LPY7	LPY6	LPT5	LPY4	LPY3	LPY2	LPY1	LPY0	Coord. Y penna ottica
21 ($15)	M7E	M6E	M5E	M4E	M3E	M2E	M1E	M0E	Abilitazione MOB
22 ($16)	--	--	RES	MCM	CSEL	X2	X1	X0	Vedi testo
23 ($17)	M7YE	M6YE	M5YE	M4YE	M3YE	M2YE	M1YE	M0YE	Espansione Y del MOB
24 ($18)	VM13	VM12	VM11	VM10	CB13	CB12	CB11	--	Puntatori alla memoria
25 ($19)	IRQ	--	--	--	ILP	IMMC	IMBC	IRST	Registro dell’interrupt
26 ($1A)	--	--	--	--	ELP	EMMC	EMBC	ERST	Abilitazione interrupt
27 ($1B)	M7DP	M6DP	M5DP	M4DP	M3DP	M2DP	M1DP	M0DP	Priorità MOB-DATA
28 ($1C)	M7MC	M8MC	M5MC	M4MC	M3MC	M2MC	M1MC	M0MC	Sel. multicolori del MOB
29 ($1D)	M7XE	M6XE	M5XE	M4XE	M3XE	M2XE	M1XE	M0XE	Espansione X del MOB
30 ($1E)	M7M	M6M	M5M	M4M	M3M	M2M	M1M	M0M	Collisione MOB-MOB
31 ($1F)	M7D	M6D	M5D	M4D	M3D	M2D	M1D	M0D	Collisione MOB-DATA
32 ($20)	--	--	--	--	EC3	EC2	EC1	EC0	Colore esterno
33 ($21)	--	--	--	--	B0C3	B0C2	B0C1	B0C0	Colore fondo #0
34 ($22)	--	--	--	--	B1C3	B1C2	B1C1	B1C0	Colore fondo #1
35 ($23)	--	--	--	--	B2C3	B2C2	B2C1	B2C0	Colore fondo #2
36 ($24)	--	--	--	--	B3C3	B3C2	B3C1	B3C0	Colore fondo #3
37 ($25)	--	--	--	--	MM03	MM02	MM01	MM00	Nulticolore #1 del MOB
38 ($26)	--	--	--	--	MM13	MM12	MM11	MM10	Multicolore #2 del MOB
39 ($27)	--	--	--	--	M0C3	M0C2	M0C1	M0C0	Colore MOB 0
40 ($28)	--	--	--	--	M1C3	M1C2	M1C1	M1C0	Colore MOB 1
41 ($29)	--	--	--	--	M2C3	M2C2	M2C1	M2C0	Colore MOB 2
42 ($2A)	--	--	--	--	M3C3	M3C2	M3C1	M3C0	Colore MOB 3
43 ($2B)	--	--	--	--	M4C3	M4C2	M4C1	M4C0	Colore MOB 4
44 ($2C)	--	--	--	--	M5C3	M5C2	M5C1	M5C0	Colore MOB 5
45 ($2D)	--	--	--	--	M6C3	M6C2	M6C1	M6C0	Colore MOB 6
46 ($2E)	--	--	--	--	M7C3	M7C2	M7C1	M7C0	Colore MOB 7

NOTA: Una linetta indica mancanza di collegamento e vengono letti come un "1".

APPENDICE Q

IMPOSTAZIONI DI CONTROLLO AUDIO DEL COMMODORE 64

Questa utile tabella ti dà i numeri delle chiavi che devi utilizzare nei tuoi programmi audio, in base a quale delle 3 voci del Commodore 64 vuoi usare. Per impostare o regolare un controllo audio nel tuo programma BASIC, basta inserire (POKE) il numero della seconda colonna seguito da una virgola (,) ed un numero della tabella... in questo modo: POKE 54276,17 (Seleziona una forma d'onda Triangolo per la VOCE 1).
Ricorda che, prima di produrre dei suoni, devi prima mettere a punto il VOLUME. POKE 54296, seguito da un numero da 0 a 15, imposta il volume per tutte le 3 voci.
Per generare ciascuna nota musicale occorrono 2 POKE separate... per esempio POKE 54273,34: POKE 54272,75 indicano il DO basso nella scala di campionamento qui sotto. Inoltre ... non sei limitato dai numeri mostrati nelle tabelle. Se il 34 non sembra "giusto" per un DO basso, prova il 35. Per fornire un SUSTAIN o una velocità di ATTACK più alti rispetto a quelli mostrati, somma due o più numeri insieme di SUSTAIN. (Esempi: POKE 54277,96 unisce due velocità di attacco (32 e 64) per una velocità d'attacco combinata più alta ... ma ... POKE 54277,20 fornisce una velocità d'attacco bassa (16) ed una velocità media di decadimento (4).

> NdT: nella notazione anglosassone le note musicali vengono rappresentate dalle prime lettere maiuscole dell'alfabeto partendo dalla nota LA (sì, lo sappiamo… gli inglesi sono particolari su tutto…).
> Per chi non le conoscesse ecco l'uguaglianza con la notazione italiana:
>
> A = LA B = SI C = DO D = RE E = MI F = FA G = SOL
>
> Il segno # (diesis) indica il semitono successivo di quella nota. La scala musicale completa è formata da:
>
> DO, DO#, RE, RE#, MI, FA, FA#, SOL, SOL#, LA, LA#, SI
>
> Come si può vedere, quasi tutte le note hanno il loro semitono successivo, tranne il MI ed il SI.
> Anche se qui non viene indicato, le note possono essere indicate anche con il loro semitono precedente: il bemolle (ь). Con il bemolle la scala musicale diventa:
>
> DO, REь, RE, Miь, MI, FA, SOLь, SOL, LAь, LA, SIь, SI

```
+-----------------------------------------------------------------------------+
|IMPOSTAZIONE VOLUME -- LO STESSO PER TUTTE LE 3 VOCI                         |
```

CONTR. VOLUME	POKE54296	Impostazioni da 0 (spento) a 15 (massimo)

VOCE NUMERO 1

TO CONTROL THIS SETTING:	POKE THIS NUMBER:	FOLLOWED BY ONE OF THESE NUMBERS (0 to 15 ... or ... 0 to 255 depending on range)

TO PLAY A NOTE		C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
HIGH FREQUENCY	54273	34	36	38	40	43	45	48	51	54	57	61	64	68	72
LOW FREQUENCY	54272	75	85	126	200	52	198	127	97	111	172	126	188	149	169

WAVEFORM	POKE 54276	TRIANGLE 17	SAWTOOTH 33	PULSE 65	NOISE 129

PULSE RATE (Pulse Waveform)		
HI POLSE	54275	A value of 0 to 15 (for Pulse waveform only)
LO POLSE	54274	A value of 0 to 255 (for Pulse waveform only)

ATTACK/ DECAY	POKE 54277	ATK4 128	ATK3 64	ATK2 32	ATK1 16	DEC4 8	DEC3 4	DEC2 2	DEC1 1
SUSTAIN/ RELEASE	POKE 54278	SUS4 128	SUS3 64	SUS2 32	SUS1 16	REL4 8	REL3 4	REL2 2	REL1 1

VOCE NUMERO 2

TO CONTROL THIS SETTING:	POKE THIS NUMBER:	FOLLOWED BY ONE OF THESE NUMBERS (0 to 15 ... or ... 0 to 255 depending on range)

TO PLAY A NOTE		C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
HIGH FREQUENCY	54280	34	36	38	40	43	45	48	51	54	57	61	64	68	72
LOW FREQUENCY	54279	75	85	126	200	52	198	127	97	111	172	126	188	149	169

WAVEFORM	POKE 54283	TRIANGLE 17	SAWTOOTH 33	PULSE 65	NOISE 129

PULSE RATE (Pulse Waveform)		
HI POLSE	54282	A value of 0 to 15 (for Pulse waveform only)
LO POLSE	54281	A value of 0 to 255 (for Pulse waveform only)

ATTACK/ DECAY	POKE 54284	ATK4 128	ATK3 64	ATK2 32	ATK1 16	DEC4 8	DEC3 4	DEC2 2	DEC1 1
SUSTAIN/ RELEASE	POKE 54285	SUS4 128	SUS3 64	SUS2 32	SUS1 16	REL4 8	REL3 4	REL2 2	REL1 1

VOCE NUMERO 3

TO CONTROL THIS SETTING:	POKE THIS NUMBER:	FOLLOWED BY ONE OF THESE NUMBERS (0 to 15 ... or ... 0 to 255 depending on range)

TO PLAY A NOTE		C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
HIGH FREQUENCY	54287	34	36	38	40	43	45	48	51	54	57	61	64	68	72
LOW FREQUENCY	54286	75	85	126	200	52	198	127	97	111	172	126	188	149	169

WAVEFORM	POKE 54290	TRIANGLE 17	SAWTOOTH 33	PULSE 65	NOISE 129

PULSE RATE (Pulse Waveform)		
HI POLSE	54289	A value of 0 to 15 (for Pulse waveform only)
LO POLSE	54288	A value of 0 to 255 (for Pulse waveform only)

```
+--------------+---------+------+------+------+------+-----+-----+-----+-----+
|ATTACK/       |  POKE   | ATK4 | ATK3 | ATK2 | ATK1 | DEC4| DEC3| DEC2| DEC1|
|        DECAY |  54291  | 128  |  64  |  32  |  16  |  8  |  4  |  2  |  1  |
+--------------+---------+------+------+------+------+-----+-----+-----+-----+
|SUSTAIN/      |  POKE   | SUS4 | SUS3 | SUS2 | SUS1 | REL4| REL3| REL2| REL1|
|       RELEASE|  54292  | 128  |  64  |  32  |  16  |  8  |  4  |  2  |  1  |
+--------------+---------+------+------+------+------+-----+-----+-----+----+
```

Per simulare alcuni strumenti, prova queste impostazioni:

Strumento	**Forma d'onda**	**A/D**	**S/R**	**Campo d'impulso**
Pianoforte	Pulsante	9	0	Hi-0, Lo-255
Flauto	Triangolo	96	0	Non applicabile
Arpa	Dente di sega	9	0	Non applicabile
Xilofono	Triangolo	9	0	Non applicabile
Organo	Triangolo	0	240	Non applicabile
Colliape (?)	Triangolo	0	240	Non applicabile
Fisarmonica	Triangolo	102	0	Non applicabile
Tromba	Dente di sega	96	0	Non applicabile

SIGNIFICATI DEI TERMINI AUDIO

ADSR	-- Attacco/Decadimento/Sostegno/Rilascio
Attack	-- L'andamento del suono si alza fino al valore del volume
Decay	-- L'andamento del suono decade dal valore del volume al livello del Sostegno
Sustain	-- Prolunga l'andamento con un certo volume
Release	-- L'andamento al cui volume decade dal livello di Sostegno
Waveform	-- "Forma" dell'onda sonora
Pulse	-- Qualità del tono dell'impulso della forma d'onda

NOTA: Le impostazioni di Attacco/Decadimento e Sostegno/Rilascio dovranno sempre essere inserite (POKE) nel tuo programma PRIMA di inserire (POKE) la forma d'onda.

APPENDICE R

CARATTERISTICHE DEL MICROCIRCUITO 6581 SOUND INTERFACE DEVICE (SID)

CONCETTO

Il Sound Device Interface 6581 (SID) è un unico microcircuito, sintetizzatore elettronico a 3 voci musicali/generatore di effetti audio, compatibile con il 6510 e famiglia di microprocessori similari. Il SID fornisce un ampia gamma, controllo dell'alta risoluzione del timbro (frequenza), timbrica del tono (contenuto armonico) e dinamica (volume). La circuiteria di controllo specializzato minimizza l'impiego di software, facilita l'uso per i giochi arcade/home-video e di strumenti musicali.

CARATTERISTICHE

- 3 OSCILLATORI DI TONALITÀ
  Campo: da 0 a 4 kHz

- 4 FORME D'ONDA PER OGNI OSCILLATORE
  Triangolare, Dente di sega, Impulso Variabile, Rumore
- 3 MODULATORI D'AMPIEZZA
  Campo: 48 dB
- 3 GENERATORI DI INVILUPPO
  Risposta esponenziale
  Velocità di Attacco: da 2 ms a 8 s
  Velocità di Decadimento: da 6 ms a 24 s
  Livello di Sostegno: da 0 al valore del volume
  Velocità di Rilascio: da 6 ms a 24 s
- SINCRONIZZAZIONE DEGLI OSCILLATORI
- MODULAZIONE AD ANELLO

DESCRIZIONE

Il 6581 comprende tre "voci" sinterizzate che possono essere utilizzate indipendentemente o insieme a qualsiasi altra (o sorgenti audio esterne) per creare suoni complessi. Ciascuna voce comprende un Oscillatore di Tono/Generatore di forma d'onda, un Generatore di Inviluppo ed un Modulatore d'Ampiezza. L'Oscillatore di Tono controlla l'altezza della voce per un'ampia gamma. L'Oscillatore presenta quattro forme d'onda su una data frequenza, con il contenuto armonico unico di ciascuna forma d'onda, fornendo un semplice controllo sul timbro del tono. Le dinamiche di volume dell'oscillatore sono controllate dal Modulatore d'Ampiezza sotto la direzione del Generatore di Inviluppo. Quando attivato, il Generatore di Inviluppo crea un inviluppo d'ampiezza a velocità programmabile che incrementa e decrementa il volume. Oltre alle tre voci, il chip fornisce un Filtro programmabile per generare timbri complessi e dinamici del tono attraverso sintesi sottattive.
Il SID permette al microprocessore di leggere l'uscita che varia da un terzo Oscillatore e da un terzo Generatore di Inviluppo. Queste uscite possono essere utilizzate come sorgente di informazioni della modulazione per creare vibrato, frequenza/curve di filtro ed effetti similari. Il terzo oscillatore può agire anche come generatore di numeri casuali per i giochi. Per l'interfacciamento del SID con potenziometri sono presenti due convertitori A/D. Questi possono essere usati come "paddle" in un ambiente di gioco oppure come controlli di un pannello di un sintetizzatore musicale. Il SID può elaborare segnali audio esterni, permettendo la concatenazione di più SID o la miscelazione in complessi sistemi polifonici.

REGISTRI DI CONTROLLO DEL SID

Per controllare la generazione dei suoni, nel SID ci sono 29 registri ad otto bit. Questi registri sono o a sola scrittura o a sola lettura e sono elencati qui sotto, nella Tabella 1.

SS = solo scrittura
SL = solo lettura

Tabella 1. Mappa dei registri del SID

Reg# D H	D7	D6	D D5	A D4	T D3	A D2	D1	D0	Nome registro	Tipo reg
									Voce 1	
0 00	F7	F6	F5	F4	F3	F2	F1	F0	Frequenza bassa	SS
1 01	F15	F14	F13	F12	F11	F10	F9	F8	Frequenza alta	SS
2 02	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0	PW basso	SS
3 03	--	--	--	--	PW11	PW10	PW9	PW8	PW alto	SS
4 04	rumore	impulso	sega	triang	test	anello	sincro	porta	Registro controllo	SS
5 05	ATK3	ATK2	ATK1	ATK0	DCY3	DCY2	DCY1	DCY0	Attacco/Decadimento	SS
6 06	STN3	STN2	STN1	STN0	RLS3	RLS2	RLS1	RLS0	Sostegno/Rilascio	SS
									Voce 2	
7 07	F7	F6	F5	F4	F3	F2	F1	F0	Frequenza bassa	SS

8	08	F15	F14	F13	F12	F11	F10	F9	F8	Frequenza alta	SS
9	09	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0	PW basso	SS
10	0A	--	--	--	--	PW11	PW10	PW9	PW8	PW alto	SS
11	0B	rumore	impulso	sega	triang	test	anello	sincro	porta	Registro controllo	SS
12	0C	ATK3	ATK2	ATK1	ATK0	DCY3	DCY2	DCY1	DCY0	Attacco/Decadimento	SS
13	0D	STN3	STN2	STN1	STN0	RLS3	RLS2	RLS1	RLS0	Sostegno/Rilascio	SS
										Voce 3	
14	0E	F7	F6	F5	F4	F3	F2	F1	F0	Frequenza bassa	SS
15	0F	F15	F14	F13	F12	F11	F10	F9	F8	Frequenza alta	SS
16	10	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0	PW basso	SS
17	11	--	--	--	--	PW11	PW10	PW9	PW8	PW alto	SS
18	12	rumore	impulso	sega	triang	test	anello	sincro	porta	Registro controllo	SS
19	13	ATK3	ATK2	ATK1	ATK0	DCY3	DCY2	DCY1	DCY0	Attacco/Decadimento	SS
20	14	STN3	STN2	STN1	STN0	RLS3	RLS2	RLS1	RLS0	Sostegno/Rilascio	SS
										Filtro	
21	15	--	--	--	--	--	FC2	FC1	FC0	FC basso	SS
22	16	FC10	FC9	FC8	FC7	FC6	FC5	FC4	FC3	FC alto	SS
23	17	RES3	RES2	RES1	RES0	FILTEX	FILT3	FILT2	FILT1	Reset/Filtro	SS
24	18	3OFF	HP	BP	LP	VOL3	VOL2	VOL1	VOL0	Modalità/Volume	SS
										Varie	
25	19	PX7	PX6	PX5	PX4	PX3	PX2	PX1	PX0	POT X	SL
26	1A	PY7	PY6	PY5	PY4	PY3	PY2	PY1	PY0	POT Y	SL
27	1B	O7	O6	O5	O4	O3	O2	O1	O0	Oscill3/Casuale	SL
28	1C	E7	E6	E5	E4	E3	E2	E1	E0	Inviluppo3	SL

APPENDICE S

COMANDI ED ISTRUZIONI PER DISCO E STAMPANTE

I successivi comandi ed istruzioni BASIC ti permettono di eseguire una vasta gamma di operazioni sui gestori disco e qualsiasi stampante compatibile Commodore.

CLOSE

TIPO: Istruzione di I/O
FORMATO: CLOSE <numero file>

Azione: Questa istruzione chiude qualsiasi file di dati o canale su un dispositivo. Il numero del file è lo stesso di quando il file o il dispositivo è stato aperto (vedi istruzione OPEN ed il paragrafo sulla programmazione di INPUT/OUTPUT).

Durante la lavorazione su dispositivi di memorizzazione, come i dischi, l'effetto di CLOSE memorizza sul dispositivo tutte le memorie intermedie incomplete. Se l'operazione non viene eseguita, il file sul disco risulterà illeggibile. L'operazione di CLOSE non è necessaria con altri dispositivi, ma libera più memoria per altri file. Per maggiori dettagli consulta il manuale del tuo dispositivo esterno.

ESEMPI dell'istruzione CLOSE:

```
10 CLOSE 1
20 CLOSE X
30 CLOSE 9 * (1 + J)
```

CMD

TIPO: Istruzione di I/O
FORMATO: CMD <numero file>[,stringa]

Azione: Questa istruzione scambia il dispositivo d'uscita principale: dallo schermo della TV al file indicato. Questo file potrebbe trovarsi su disco, stampante o un dispositivo di I/O come il modem. Il numero del file deve essere specificato in una precedente istruzione OPEN. La stringa, quando indicato, viene inviata al file. L'operazione è utile per intestare stampati, ecc.

Quando questo comando è in funzione, qualsiasi istruzione PRINT e comando LIST non verranno visualizzati sullo schermo ma il testo verrà inviato al file nello stesso formato.
Per ri-direzionare l'uscita sullo schermo, prima di chiudere (CLOSE) il comando PRINT# dovrà inviare una riga vuota al dispositivo CMD, in modo che smetta di aspettare dati (detto "dispositivo un-listening").
Qualsiasi errore di sistema (come ?SYNTAX ERROR) farà in modo che l'uscita ritorni allo schermo. Poiché dopo una condizione d'errore i dispositivi non diventano un-listening, dovrai inviare una riga vuota. (Per ulteriori dettagli consulta il manuale della tua stampante o gestore disco).

ESEMPI dell'istruzione CMD:

```
OPEN 4,4: CMD 4,"TITOLO": LIST: REM LISTA IL PROGRAMMA SULLA STAMPANTE
PRINT#4: CLOSE 4: REM RENDE SORDA E CHIUDE LA STAMPANTE
```

```
10 OPEN 1, 8 ,4, "TEST": REM CREA UN FILE SEQUENZIALE
20 CMD 8: REM USCITA SUL FILE, NON SU SCHERMO
30 FOR L = 1 TO 100
40 PRINT L: REM INSERISCE IL NUMERO SUL BUFFER DEL DISCO
50 NEXT
60 PRINT# 1: REM RENDE SORDO
70 CLOSE 1: REM SCRIVE IL BUFFER INCOMPLETO, TERMINA CORRETTAMENTE
```

GET#

TIPO: Istruzione di I/O
FORMATO: GET# <numero file>,<lista la variabile>

Azione: Questa istruzione legge un carattere alla volta dal dispositivo o dal file indicato. In pratica opera come l'istruzione GET, ma il dato proviene da un posto diverso dalla tastiera. Se non viene ricevuto alcun carattere, la variabile viene impostata su una stringa vuota (uguale a"") oppure a 0 in caso di variabili numeriche. I caratteri usati per separare dati nei file, come la virgola (,) o il codice del tasto <INVIO> (codice ASC 13), vengono ricevuti come qualsiasi altro carattere.

Quando usato con il dispositivo #3 (schermo TV), questa istruzione leggerà i caratteri uno ad uno dallo schermo. Ciascun utilizzo di GET# sposterà il cursore di 1 posizione verso destra. Il carattere in fondo alla riga logica verrà modificato con un CHR$ (13), il codice del tasto <INVIO>.

ESEMPI dell'istruzione GET#:

```
 5 GET# 1, A$
10 OPEN 1, 3: GET# 1, Z7$
20 GET# 1, A, B, C$, D$
```

INPUT#

TIPO: Istruzione di I/O
FORMATO: INPUT# <numero file>,<lista la variabile>

Azione: Di solito questo è il modo più veloce e più facile per recuperare dati memorizzati in un file su disco. Il dato si trova nella forma di variabili complete, lunghe fino a 80 caratteri, ed è all'opposto del metodo uno alla volta di GET#. Prima di tutto il file deve essere aperto (OPEN), poi INPUT# può riempire le variabili.

Il comando INPUT# ritiene una variabile completa quando legge un codice di <INVIO> (CHR$ (13)), una virgola (,), un punto e virgola (;) o due punti (:). Nel caso in cui questi caratteri fossero necessari durante la scrittura, sarà possibile racchiuderli tra virgolette (vedi l'istruzione PRINT#).
Se il tipo di variabile usato è numerico e vengono ricevuti caratteri non-numerici, ne risulterà un errore di BAD DATA. INPUT# può leggere stringhe lunghe fino a 80 caratteri; una stringa oltre questo limite ne risulterà un errore TOO LONG.
Quando usato con il dispositivo #3 (lo schermo), questa istruzione leggerà un'intera riga logica e sposterà il cursore in basso, sulla riga successiva.

ESEMPI dell'istruzione INPUT#:

```
10 INPUT# 1, A
20 INPUT# 2, A$, B$
```

LOAD

TIPO: Comando
FORMATO: LOAD "<nome-file>",<dispositivo>[,<indirizzo>]

Azione: L'istruzione LOAD legge i contenuti di un file di programma dal disco alla memoria. In questo modo puoi utilizzare le informazioni caricate precedentemente o eventualmente modificarle. L'unità del disco è generalmente il numero del dispositivo 8. LOAD chiude tutti i file aperti e, se utilizzato in modalità diretta, prima di leggere il programma esegue un CLR (pulizia schermo). Se LOAD viene eseguito all'interno di un programma, verrà eseguito (RUN) detto programma. Questo significa che puoi utilizzare LOAD per "concatenare" insieme più programmi. Durante un'operazione di concatenamento nessuna variabile verrà vuotata.
Se per nome-file usi un modello di accoppiamento, verrà caricato il primo file che sia uguale al modello. L'asterisco fra virgolette, ("*"), fa in modo che venga caricato il primo nome-file della cartella del disco. Se il nome-file usato non esiste oppure se non è un file di programma, si verificherà il messaggio BASIC d'errore ?FILE NOT FOUND.
Se metti a 1 l'indirizzo secondario, farà in modo che il programma carichi nella locazione di memoria in cui era stato salvato.

ESEMPI del comando LOAD:

```
LOAD A$,8                   (Come nome-file usa il nome in A$)
LOAD "*",8                  (Carica il primo programma del disco)
LOAD "$",8                  (Carica la cartella del disco)

LOAD "FUN",8                (Carica un dato file da disco)
SEARCHING FOR FUN
LOADING
READY.

LOAD "GAME ONE",8,1         (Carica un file su una specifica locazione di memoria
SEARCHING FOR GAME ONE       in cui era stato salvato il programma su disco)
LOADING
READY.
```

OPEN

TIPO: Istruzione di I/O
FORMATO: OPEN <numero-file>,<dispositivo>[,<indirizzo>][,"<nome-file>[,<tipo>]
[,<modalità>]"]

Azione: Questa istruzione apre un canale, per l'ingresso e/o l'uscita, su un dispositivo periferico. Per ogni istruzione OPEN non c'è bisogno di mettere tutte quelle voci. Alcune istruzioni OPEN richiedono solo 2 voci:

1) IL NUMERO DEL FILE LOGICO

2) IL NUMERO DEL DISPOSITIVO

<numero-file> è il numero del file logico al quale si riferiscono le istruzioni di OPEN, CLOSE, CMD, GET#, INPUT# e PRINT# con ciascun altro, e li associa al nome-file ed al pezzo di apparecchiatura da utilizzare. Il numero del file logico può variare da 1 a 255 e puoi assegnare ad esso qualsiasi numero che rientri in questo campo.

> NOTA: In realtà i numeri di file sopra il 128 erano stati progettato per altri scopi; come numeri di file è una buona abitudine utilizzare solo numeri al di sotto di 127

Ciascun dispositivo periferico (stampante, gestore disco) nel sistema possiede un suo numero proprio al quale risponde. Il numero <dispositivo> viene usato con OPEN per indicare su quale dispositivo si trova il file di dati. Anche le unità periferiche come i gestori disco o le stampanti rispondono a diversi indirizzi secondari. Pensa a questi come codici che dicono ad ogni dispositivo quale operazione eseguire. Il numero del file logico del dispositivo viene usato con tutti i GET#, INPUT# e PRINT#. Il nome-file può anche essere tralasciato, ma in un secondo tempo, nel tuo programma, se non l'hai dato NON puoi richiamare il file per nome. Per i file su disco gli indirizzi secondari, da 1 a 14, sono disponibili per i file di dati, ma altri numeri hanno dei significati speciali nei comandi del DOS. Quando ti servi del gestore disco devi utilizzare un indirizzo secondario. (Vedi i dettagli sui comandi DOS del manuale del tuo gestore disco).
Il <nome-file> è una stringa di 1-16 caratteri ed è facoltativo per i file da stampante. Se il <tipo> viene omesso, il tipo di file verrà automaticamente assunto come file programma a meno che venga data la <modalità>. I file sequenziali vengono aperti in lettura <modalità>= R a meno che non venga indicato quel file con apertura in scrittura <modalità>=W. Un <tipo> di file può essere usato per aprire un file relativo esistente; in questo caso per <tipo> usa REL. I file relativi e sequenziali sono solamente per il disco.
Se cerchi di accedere ad un file prima che venga aperto, si verificherà il messaggio BASIC d'errore ?FILE NOT OPEN. Se cerchi di aprire un file in lettura che non esiste, si verificherà il messaggio BASIC d'errore ?FILE NOT FOUND. Se viene aperto un file in scrittura e nome-file esiste già sul disco, si verificherà il messaggio DOS d'errore FILE EXISTS. Se viene aperto un file già aperto, si verificherà il messaggio BASIC d'errore FILE OPEN. (Per altri particolari consulta il manuale della stampante).

ESEMPI delle istruzioni OPEN:

10 OPEN 2,8,4,"DISK-OUTPUT,SEQ,W"	(Apre in scrittura su disco un file sequenz.)
10 OPEN 50,0	(Ingresso da tastiera)
10 OPEN 12,3	(Uscita su schermo)
10 OPEN 130,4	(Uscita su stampante)
10 OPEN 1,2,0,CHR$(10)	(Apre un canale per il dispositivo RS-232)
10 OPEN 1,4,0,"STRINGA"	(Invia maiuscolo/grafica sulla stampante)
10 OPEN 1,4,7,"STRINGA"	(Invia maiuscolo/minuscolo alla stampante)
10 OPEN 1,5,7,"STRINGA"	(Invia maiuscolo/minuscolo alla stampante con dispositivo #5)
10 OPEN 1,8,15,"COMANDO"	(Invia un comando al disco)
10 OPEN 1,8,1,"NOME,L"+CHR$(X)	(Apre il file relativo (la prima volta) dove X è la lunghezza del record relativo)
10 OPEN 1,8,1,"NOME"	(Legge file relativo o sequenziale)

PRINT#

TIPO: Istruzione di I/O
FORMATO: PRINT# <numero-file>[,<variabile>][<,/;><variabile>]...

Azioni: L'istruzione PRINT# si usa per scrivere elementi di dati su un file logico, e deve utilizzare lo stesso numero usato per aprire il file. L'uscita passa al numero del dispositivo usato nell'istruzione OPEN. Le espressioni <variabile>, nelle voci

elencate, possono essere di qualsiasi tipo. I caratteri della punteggiatura tra gli elementi sono uguali all'istruzione PRINT e possono essere usati allo stesso modo. Gli effetti della punteggiatura sono differenti in due modi significativi.
Se alla fine dell'elenco non c'è alcuna punteggiatura, in fondo ai dati viene scritto un ritorno di carrello ed un avanzamento di riga. Se l'elenco d'uscita termina con una virgola o punto e virgola, viene omesso il ritorno di carrello e l'avanzamento di riga. Indipendentemente dalla punteggiatura, la successiva istruzione PRINT# inizia l'uscita nella prossima posizione del carattere disponibile. Quando si usa l'istruzione INPUT#, l'avanzamento di riga agirà da interruzione lasciando una variabile vuota quando verrà eseguita la successiva istruzione INPUT#. L'avanzamento di riga può essere soppresso o compensato come mostrato nell'esempi qui sotto.

Il modo più facile per scrivere più di una variabile su un file su disco, è impostare una variabile di stringa con CHR$ (13) ed utilizzare detta stringa in mezzo a tutte le altre variabili durante la scrittura del file.

ESEMPI di istruzione PRINT#:

```
1)
10 OPEN 1,8,4, "MY FILE"
20 R$=CHR$(18)                        (Sostituendo CHR$(13) in
30 PRINT#1,1;R$;2;R$;3;R$;4;R$;5      CHR$(44) metti una "," tra ogni
40 PRINT#1,6                          variabile. CHR$(59) metterà un ";"
50 PRINT#1,7                          tra ciascuna variabile)
2)
10 CO$=CHR$(44): CR$=CHR$(13)
20 PRINT#1, "AAA"CO$"BBB",            AAA,BBB  CCCDDDEEE
   "CCC";"DDD";"EEE"CR$               (ritorno di carrello)
   "FFF"CR$;                          FFF(ritorno di carrello)
30 INPUT#1, A$,BCDE$,F$

3)
 5 CR$=CHR$(13)
10 PRINT#2, "AAA";CR$;"BBB"           (10 spazi vuoti) AAA
20 PRINT#2, "CCC";                    BBB
                                      (10 spazi vuoti) CCC
30 INPUT#2, A$,B$,DUMMY$,C$
```

SAVE

TIPO: Comando
FORMATO: SAVE "<nome-file>",<numero-dispositivo>[,<indirizzo>]

Azione: Il comando SAVE serve a memorizzare su un file del disco il programma che attualmente si trova in memoria. Il programma da salvare viene interessato dal comando solo mentre avviene il salvataggio. Dopo che è stata completata l'operazione SAVE, il programma rimane nella memoria del computer fino a quando non venga inserito qualcos'altro con un altro comando. Il tipo di file sarà "prg" (programma). L'istruzione SAVE può essere usata anche in un programma; in questo caso l'esecuzione continuerà con l'istruzione successiva dopo aver completato il SAVE.
Durante il salvataggio dei programmi su un disco, <nome-archivio> deve essere presente.

ESEMPI del comando SAVE:

```
SAVE "FUN DISK",8          (Salva su disco (il dispositivo 8 è il disco))
SAVE A$,8                  (Memorizza su disco con il nome in A$)
```

SAVE e REPLACE

TIPO: Comando
FORMATO: SAVE "@0:<nome-file>",<numero-dispositivo>

Azione: Questa versione del comando SAVE serve a sovrascrivere un file o programma esistente su disco. Salvando un programma con la versione regolare del comando SAVE, utilizzando un nome-file già esistente, non si memorizzerà il programma anche se non verrà indicato nessun errore del disco.

ESEMPI di SAVE e REPLACE:

SAVE "@0:ME",8 (Sovrascrive "ME" su disco con una versione aggiornata)

VERIFY

TIPO: Comando
FORMATO: VERIFY "<nome-file>",<dispositivo>

Azione: Il comando VERIFY viene utilizzato, in modalità diretta o programma, per confrontare il contenuto di un file di programma BASIC su disco con il programma attualmente in memoria. Di solito VERIFY viene utilizzato dopo un SAVE per assicurarsi che il programma sia stato memorizzato correttamente su nastro o disco.
Per i file su disco (numero del dispositivo 8), nome-file deve essere presente. Nel caso venga trovata una qualsiasi differenza nel testo del programma, verrà visualizzato il messaggio BASIC d'errore ?VERIFY ERROR. Il nome del programma può essere dato sia fra virgolette ("") che come variabile di stringa.

ESEMPI di comando VERIFY:

```
9000 SAVE "ME",8
9010 VERIFY "ME",8
```
(Esamina il programma ME sul dispositivo 8)

SCHEDA PER RIFERIMENTO RAPIDO DEL COMMODORE 64

VARIABILI SEMPLICI

Tipo	Nome	Campo
Reale	XY	+/-1.70141183E+38 +/-2.93873588E-39
Intera	XY%	+32767...-32768
Stringa	XY$	Da 0 a 255 caratteri

X è una lettera (da A a Z); Y è una lettera (da A a Z) o un numero (da 0 a 9).
I nomi delle variabili possono avere più di 2 caratteri, ma vengono riconosciuti solo i primi due.

VARIABILI DI MATRICE

Tipo	Nome
Mono-dimensionale	XY(5)
Bi-dimensionale	XY(5,5)
Tri-dimensionale	XY(5,5,5)

È possibile utilizzare vettori fino ad undici elementi (sottocodici 0-10) dove necessario. I vettori con più di undici elementi devono essere DIMensionati.

OPERAZIONI ALGEBRICHE

Tipo	Nome
=	Assegna un valore alla variabile
-	Negativo
^	Esponenziazione
*	Moltiplicazione
/	Divisione
+	Addizione (somma)
-	sottrazione

OPERATORI RELAZIONALI E LOGICI

Tipo	Nome
=	Uguale a
<>	Non uguale a (diverso da)
<	Minore di
>	Maggiore di
<=	Minore di o Uguale a
>=	Maggiore di o Uguale a
NOT	Negazione logica
AND	"E" logico
OR	"O" logico

Le espressioni equivalgono a -1 quando vere, a 0 se false.

COMANDI DI SISTEMA

LOAD "NOME"	Carica un programma da nastro
SAVE "NOME"	Salva un programma su nastro
LOAD "NOME",8	Carica un programma su disco
SAVE "NOME",8	Salva un programma su disco
VERIFY "NOME"	Verifica se quel programma è stato salvato senza errori
RUN	Esegue un programma
RUN xxx	Esegue un programma a partire dalla riga xxx
STOP	Arresta l'esecuzione
END	Termina l'esecuzione
CONT	Prosegue l'esecuzione del programma dalla riga in cui era stato arrestato
PEEK(X)	Ritorna il contenuto della locazione di memoria X
POKE X,Y	Modifica il contenuto della locazione X con il valore Y
SYS xxxxx	Salta all'esecuzione di un programma in linguaggio macchina che inizia da xxxxx
WAIT X,Y,Z	Il programma aspetta che il contenuto della locazione X, quando operato in OR con Z ed in AND con Y, sia non zero.
USR(X)	Passa il valore di X ad una subroutine in linguaggio macchina

COMANDI DI EDITAZIONE E FORMATTAZIONE

LIST	Elenca tutto il programma
LIST A-B	Elenca il programma dalla riga A alla riga B (compresi)
LIST A-	Elenca il programma iniziando da A (compreso)
LIST –A	Elenca il programma fino ad A (compreso)
LIST A	Elenca solo A
REM Nota	Qui è possibile inserire un messaggio di commento che verrà ignorato durante l'esecuzione del programma

TAB(X)	Usato nell'istruzione PRINT. Spazia di X posizioni sullo schermo
SPC(X)	Stampa (PRINT) X spazi vuoti sulla riga
POS(X)	Ritorna l'attuale posizione del cursore
CLR/HOME	Posiziona il cursore nell'angolo superiore sinistro dello schermo
SHIFT CLR/HOME	Pulisce lo schermo e mette il cursore in posizione "Home"
SHIFT INS/DEL	Inserisce spazio nella posizione attuale del cursore
INST/DEL	Cancella un carattere nella posizione attuale del cursore
CTRL	Quando premuto con i tasti numerici di colore, seleziona il colore del testo. Si può utilizzare nell'istruzione PRINT.
Tasti CRSR	Spostano il cursore su, giù, a destra e sinistra sullo schermo
Tasto Commodore	Quando usato con SHIFT, seleziona le modalità maiuscola/minuscola e la visualizzazione grafica. Quando usato con i tasti numerici del colore, seleziona il colore facoltativo del testo

MATRICI E STRINGHE

DIM A(X,Y,Z)	Imposta per A i sottocodici massimi; si riserva spazio per gli elementi (X+1)*(Y+1)*(Z+1) iniziando da A(0,0,0)
LEN(X$)	Ritorna la quantità di caratteri in X$
STR$(X)	Ritorna il valore numerico di X covertito in stringa
VAL(X$)	Ritorna il valore numerico di X$ fino al primo carattere non numerico
CHR$(X)	Ritorna il carattere ASCII il cui codice è X
ASC(X$)	Ritorna il codice ASCII del primo carattere di X$
LEFT$(A$,X)	Ritorna gli X caratteri più a sinistra di A$
RIGHT$(A$,X)	Ritorna gli X caratteri più a destra di A$
MID$(A$,X,Y)	Ritorna Y caratteri di A$ iniziando dal carattere X

COMANDI DI ENTRATA/USCITA

INPUT A$ o A	Stampa (PRINT) un '?' sullo schermo e rimane in attesa che l'utente immetta una stringa o un valore
INPUT "ABC";A	Stampa (PRINT) un messaggio ed attende che l'utente immetta un valore. Anche come INPUT A$
GET A$ o A	Attende che l'utente digiti il valore di un carattere; non è necessario premere <RETURN>
DATA A,"B",C	Inizializza una serie di valori da utilizzare con l'istruzione READ.
READ A$ o A	Assegna ad A$ o A il valore successivo di DATA
RESTORE	Resetta il puntatore dei dati per leggere ancora l'elenco dei DATA
PRINT "A=";A	Stampa (PRINT) la string 'A=' ed il valore di A; il punto e virgola (';') elimina gli spazi; la virgola (',') tabula il dato nel campo successivo

FLUSSO DEL PROGRAMMA

GOTO X	Salta alla riga X proseguendo le istruzioni da qui
IF A=3 THEN 10	***SE*** l'affermazione è vera ***ALLORA*** esegue la parte seguente dell'istruzione. ***SE*** falsa, esegue il numero di riga successivo
FOR A=1 TO 10 STEP 2: NEXT	Esegue tutte le istruzioni tra il FOR ed il rispettivo NEXT con A che inizia da 1 e termina a 10 contando 2 valori alla volta. Se STEP non viene indicato il conteggio avanza di 1 valore alla volta
NEXT A	Stabilisce la fine del ciclo. A è facoltativa
GOSUB 2000	Salta ad una sotto routine che inizia dalla riga 2000
RETURN	Indica la fine della sotto routine e ritorna all'istruzione che segue l'ultima istruzione GOSUB
ON X GOTO A,B	Salta al numero di riga dell'elenco in base al valore di X. Se X=1 salta su A, ecc.
ON X GOSUB A,B	Salta alla sotto routine dell'elenco in base al valore di X.

QUALCOSA SULLA GUIDA PER GLI UTENTI DEL COMMODORE 64...

Colore di rilievo... sintesi del suono... grafica... capacità di calcolo... l'unione sinergica di tecnologie allo stato-dell'arte. Queste caratteristiche rendono il Commodore 64 il computer personale più all'avanguardia nella sua classe.

La Guida dell'Utente del COMMODORE 64 ti aiuta ad avvicinarti ai calcolatori, anche se prima non hai mai utilizzato un computer. In modo assai chiaro, con istruzioni passo-per-passo, ti viene data più che un'infarinatura nel linguaggio BASIC e come poter inserire il Commodore 64 in una miriade di impieghi.

Per quelli che hanno già familiarità con i microcomputer, i paragrafi sulla programmazione avanzata e le appendici spiegano le caratteristiche progredite del Commodore 64 e come ottenere la maggior parte di queste possibilità.

c o m m o d o r e
C O M P U T E R

****

Fine dell'iper-testo del Manuale dell'Utente per il Microcomputer Commodore 64 (II edizione)

****